

autoload.zsh(1)

NAME

autoload.zsh - a shell script

SYNOPSIS

Documentation automatically generated with `zsdoc'

FUNCTIONS

```
.zi-analytics-menu
.zi-any-to-uspl2
.zi-at-eval
.zi-build-module
.zi-cd
.zi-cdisable
.zi-cenable
.zi-changes
.zi-check-comp-consistency
.zi-check-which-completions-are-enabled
.zi-check-which-completions-are-installed
.zi-clear-completions
.zi-clear-report-for
.zi-compile-uncompile-all
.zi-compiled
.zi-confirm
.zi-create
.zi-delete
.zi-diff-env-compute
.zi-diff-functions-compute
.zi-diff-options-compute
.zi-diff-parameter-compute
.zi-edit
.zi-exists-message
.zi-find-completions-of-plugin
.zi-format-env
.zi-format-functions
.zi-format-options
.zi-format-parameter
.zi-get-completion-owner
.zi-get-completion-owner-uspl2col
.zi-get-path
.zi-glance
.zi-help
```

```
.zi-list-bindkeys
.zi-list-compdef-replay
.zi-ls
.zi-module
.zi-pager
.zi-prepare-readlink
.zi-recall
.zi-recently
.zi-registered-ice-mods
.zi-registered-subcommands
.zi-restore-extendedglob
.zi-run-delete-hooks
.zi-save-set-extendedglob
.zi-search-completions
.zi-self-update
.zi-show-all-reports
.zi-show-completions
.zi-show-debug-report
.zi-show-registered-plugins
.zi-show-report
.zi-show-times
.zi-show-zstatus
.zi-stress
.zi-uncompile-plugin
.zi-uninstall-completions
.zi-unload
.zi-unregister-plugin
.zi-update-all-parallel
.zi-update-or-status
.zi-update-or-status-all
.zi-update-or-status-snippet
.zi-wait-for-update-jobs
AUTOLOAD is-at-least
```

DETAILS

Script Body

Has 4 line(s). No functions are called (may set up e.g. a hook, a Zle widget bound to a key, etc.).

Uses feature(s): *source*

zi-analytics-menu

~~~~~

```
FUNCTION: .zi-analytics-menu []
Statistics, benchmarks and information.
```

User-action entry point.

Has 25 line(s). Calls functions:

```
.zi-analytics-menu  
'-- zi.zsh/+zi-message
```

Called by:

```
zi.zsh/zi
```

*zi-any-to-uspl2*

~~~~~

```
FUNCTION: .zi-any-to-uspl2 [[[  
Converts given plugin-spec to format that's used in keys for hash tables.  
So basically, creates string "user/plugin" (this format is called: uspl2).
```

```
$1 - plugin spec (4 formats: user---plugin, user/plugin, user, plugin)  
$2 - (optional) plugin (only when $1 - i.e. user - given)
```

Has 2 line(s). Calls functions:

```
.zi-any-to-uspl2  
'-- zi.zsh/.zi-any-to-user-plugin
```

Called by:

```
.zi-clear-report-for  
.zi-exists-message
```

zi-at-eval

~~~~~

```
FUNCTION: .zi-at-eval [[[
```

Has 5 line(s). Calls functions:

```
.zi-at-eval
```

```
`-- zi.zsh/@zi-substitute
```

Uses feature(s): *eval*

Not called by script or any function (may be e.g. a hook, a Zle widget, etc.).

*zi-build-module*

~~~~~

```
FUNCTION: .zi-build-module [[  
Performs ./configure && make on the module and displays information how to load  
the module in .zshrc.
```

Has 40 line(s). Calls functions:

```
.zi-build-module  
`-- zi.zsh/+zi-message
```

Called by:

```
.zi-module
```

zi-cd

~~

```
FUNCTION: .zi-cd [[  
Jumps to plugin's directory (in  $\square$  ZI  $\square$  home directory).
```

User-action entry point.

```
$1 - plugin spec (4 formats: user---plugin, user/plugin, user, plugin)  
$2 - plugin (only when $1 - i.e. user - given)
```

Has 15 line(s). Calls functions:

```
.zi-cd  
`-- zi.zsh/+zi-message
```

Uses feature(s): *setopt*

Not called by script or any function (may be e.g. a hook, a Zle widget, etc.).

zi-cdisable

~~~~~

FUNCTION: `.zi-cdisable` `[[[`  
Enables given installed completion.

User-action entry point.

`$1` - e.g. `"_mkdir"` or `"mkdir"`

Has 27 line(s). Calls functions:

`.zi-cdisable`

Called by:

`zi.zsh/zi`

## *zi-cenable*

~~~~~

FUNCTION: `.zi-cenable` `[[[`
Disables given installed completion.

User-action entry point.

`$1` - e.g. `"_mkdir"` or `"mkdir"`

Has 26 line(s). Calls functions:

`.zi-cenable`

Called by:

`zi.zsh/zi`

zi-changes

~~~~~

```
FUNCTION: .zi-changes []  
Shows `git log` of given plugin.
```

User-action entry point.

\$1 - plugin spec (4 formats: user---plugin, user/plugin, user, plugin)  
\$2 - plugin (only when \$1 - i.e. user - given)

Has 6 line(s). Calls functions:

```
.zi-changes  
|-- side.zsh/.zi-exists-physically-message  
'-- zi.zsh/.zi-any-to-user-plugin
```

Not called by script or any function (may be e.g. a hook, a Zle widget, etc.).

*zi-check-comp-consistency*

~~~~~

```
FUNCTION: .zi-check-comp-consistency []  
  ZI  creates symlink for each installed completion.  
This function checks whether given completion (i.e. file like "_mkdir") is indeed  
a symlink.  
Backup file is a completion that is disabled - has the leading "_" removed.
```

\$1 - path to completion within plugin's directory
\$2 - path to backup file within plugin's directory

Has 11 line(s). Doesn't call other functions.

Called by:

```
.zi-cdisable  
.zi-cenable
```

zi-check-which-completions-are-enabled

~~~~~

```
FUNCTION: .zi-check-which-completions-are-enabled []
```

For each argument that each should be a path to completion within a plugin's dir, it checks whether that completion is disabled - returns 0 or 1 on corresponding positions in reply.

Uninstalled completions will be reported as "0" - i.e. disabled

\$1, ... - path to completion within plugin's directory

Has 10 line(s). Doesn't call other functions.

Called by:

.zi-show-report

*zi-check-which-completions-are-installed*

~~~~~

FUNCTION: .zi-check-which-completions-are-installed [[
For each argument that each should be a path to completion within a plugin's dir, it checks whether that completion is installed - returns 0 or 1 on corresponding positions in reply.

\$1, ... - path to completion within plugin's directory

Has 11 line(s). Doesn't call other functions.

Called by:

.zi-show-report

zi-clear-completions

~~~~~

FUNCTION: .zi-clear-completions [[  
Delete stray and improper completions.

Completions live even when plugin isn't loaded - if they are installed and enabled.

User-action entry point.

Has 35 line(s). Calls functions:

```
.zi-clear-completions
|-- side.zsh/.zi-any-colorify-as-uspl2
`-- zi.zsh/+zi-message
```

Uses feature(s): *setopt*

Called by:

```
zi.zsh/.zi-prepare-home
zi.zsh/zi
```

*zi-clear-report-for*

~~~~~

FUNCTION: `.zi-clear-report-for` `[[[`
Clears all report data for given user/plugin. This is done by resetting all related global `ZI_*` hashes.

`$1` - plugin spec (4 formats: `user---plugin`, `user/plugin`, `user`, `plugin`)
`$2` - (optional) plugin (only when `$1` - i.e. `user` - given)

Has 23 line(s). Calls functions:

```
.zi-clear-report-for
```

Called by:

```
.zi-unload
additional.zsh/.zi-clear-debug-report
```

zi-compile-uncompile-all

~~~~~

FUNCTION: `.zi-compile-uncompile-all` `[[[`  
Compiles or uncompiles all existing (on disk) plugins.



User-action entry point.

Has 19 line(s). Calls functions:

```
.zi-compile-uncompile-all  
|-- install.zsh/.zi-compile-plugin  
|-- side.zsh/.zi-any-colorify-as-uspl2  
'-- zi.zsh/.zi-any-to-user-plugin
```

Uses feature(s): *setopt*

Called by:

zi.zsh/zi

*zi-compiled*

~~~~~

FUNCTION: .zi-compiled []
Displays list of plugins that are compiled.

User-action entry point.

Has 23 line(s). Calls functions:

```
.zi-compiled  
|-- side.zsh/.zi-any-colorify-as-uspl2  
|-- zi.zsh/+zi-message  
'-- zi.zsh/.zi-any-to-user-plugin
```

Uses feature(s): *setopt*

Called by:

zi.zsh/zi

zi-confirm

~~~~~

FUNCTION: .zi-confirm []

Prints given question, waits for "y" key, evals given expression if "y" obtained

\$1 - question  
\$2 - expression

Has 22 line(s). Doesn't call other functions.

Uses feature(s): *eval*, *read*

Called by:

.zi-delete

*zi-create*

~~~~~

FUNCTION: .zi-create []
Creates a plugin, also on Github (if not "_local/name" plugin).

User-action entry point.

\$1 - (optional) plugin spec (4 formats: user---plugin, user/plugin, user, plugin)
\$2 - (optional) plugin (only when \$1 - i.e. user - given)

Has 103 line(s). Calls functions:

.zi-create
|-- side.zsh/.zi-any-colorify-as-uspl2
|-- side.zsh/.zi-exists-physically
'-- zi.zsh/.zi-any-to-user-plugin

Uses feature(s): *autoload*, *setopt*, *vared*

Not called by script or any function (may be e.g. a hook, a Zle widget, etc.).

zi-delete

~~~~~

FUNCTION: .zi-delete []  
Deletes plugin's or snippet's directory (in "ZI" home directory).

User-action entry point.

\$1 - snippet URL or plugin spec (4 formats: user---plugin, user/plugin, user, plugin)  
\$2 - plugin (only when \$1 - i.e. user - given)

Has 94 line(s). Calls functions:

```
.zi-delete  
|-- side.zsh/.zi-compute-ice  
|-- zi.zsh/+zi-prehelp-usage-message  
|-- zi.zsh/.zi-any-to-user-plugin  
'-- zi.zsh/.zi-parse-opts
```

Uses feature(s): *setopt*

Not called by script or any function (may be e.g. a hook, a Zle widget, etc.).

*zi-diff-env-compute*

~~~~~

FUNCTION: .zi-diff-env-compute []
Computes ZI_PATH, ZI_FPATH that hold (f)path components
added by plugin. Uses data gathered earlier by .zi-diff-env().

\$1 - user/plugin

Has 28 line(s). Doesn't call other functions.

Uses feature(s): *setopt*

Called by:

```
.zi-show-report  
.zi-unload
```

zi-diff-functions-compute

~~~~~

FUNCTION: .zi-diff-functions-compute []  
Computes FUNCTIONS that holds new functions added by plugin.

Uses data gathered earlier by `.zi-diff-functions()`.

\$1 - user/plugin

Has 16 line(s). Doesn't call other functions.

Uses feature(s): *setopt*

Called by:

`.zi-show-report`  
`.zi-unload`

*zi-diff-options-compute*

~~~~~

FUNCTION: `.zi-diff-options-compute` []
Computes `OPTIONS` that holds options changed by plugin.
Uses data gathered earlier by `.zi-diff-options()`.

\$1 - user/plugin

Has 16 line(s). Doesn't call other functions.

Uses feature(s): *setopt*

Called by:

`.zi-show-report`
`.zi-unload`

zi-diff-parameter-compute

~~~~~

FUNCTION: `.zi-diff-parameter-compute` []  
Computes `ZI_PARAMETERS_PRE`, `ZI_PARAMETERS_POST` that hold  
parameters created or changed (their type) by plugin. Uses  
data gathered earlier by `.zi-diff-parameter()`.

\$1 - user/plugin

Has 27 line(s). Doesn't call other functions.

Uses feature(s): *setopt*

Called by:

```
.zi-show-report  
.zi-unload
```

*zi-edit*

~~~~

```
FUNCTION: .zi-edit [[  
Runs $EDITOR on source of given plugin. If the variable is not set then defaults  
to 'code'.
```

User-action entry point.

```
$1 - plugin spec (4 formats: user---plugin, user/plugin, user, plugin)  
$2 - plugin (only when $1 - i.e. user - given)
```

Has 19 line(s). Calls functions:

```
.zi-edit  
'-- side.zsh/.zi-compute-ice
```

Not called by script or any function (may be e.g. a hook, a Zle widget, etc.).

zi-exists-message

~~~~~

```
FUNCTION: .zi-exists-message [[  
Checks if plugin is loaded. Testable. Also outputs error message if plugin is not  
loaded.
```

```
$1 - plugin spec (4 formats: user---plugin, user/plugin, user, plugin)  
$2 - (optional) plugin (only when $1 - i.e. user - given)
```

Has 7 line(s). Calls functions:

```
.zi-exists-message
```

```
`-- side.zsh/.zi-any-colorify-as-uspl2
```

Called by:

```
.zi-show-report  
.zi-unload
```

### *zi-find-completions-of-plugin*

~~~~~

FUNCTION: `.zi-find-completions-of-plugin` `[[`
Searches for completions owned by given plugin.
Returns them in 'reply' array.

\$1 - plugin spec (4 formats: user---plugin, user/plugin, user, plugin)
\$2 - plugin (only when \$1 - i.e. user - given)

Has 5 line(s). Calls functions:

```
.zi-find-completions-of-plugin  
`-- zi.zsh/.zi-any-to-user-plugin
```

Uses feature(s): *setopt*

Called by:

```
.zi-show-report
```

zi-format-env

~~~~~

FUNCTION: `.zi-format-env` `[[`  
Creates one-column text about FPATH or PATH elements added when given plugin was loaded.

\$1 - user/plugin (i.e. uspl2 format of plugin-spec)  
\$2 - if 1, then examine PATH, if 2, then examine FPATH

Has 15 line(s). Doesn't call other functions.

Called by:

```
.zi-show-report
```

### *zi-format-functions*

~~~~~

```
FUNCTION: .zi-format-functions []  
Creates a one or two columns text with functions created by given plugin.
```

```
$1 - user/plugin (i.e. uspl2 format of plugin-spec)
```

Has 34 line(s). Doesn't call other functions.

Called by:

```
.zi-show-report
```

zi-format-options

~~~~~

```
FUNCTION: .zi-format-options []  
Creates one-column text about options that changed when plugin "$1" was loaded.
```

```
$1 - user/plugin (i.e. uspl2 format of plugin-spec)
```

Has 19 line(s). Calls functions:

```
.zi-format-options
```

Called by:

```
.zi-show-report
```

### *zi-format-parameter*

~~~~~

```
FUNCTION: .zi-format-parameter []  
Creates one column text that lists global parameters that changed when the given  
plugin was loaded.
```

\$1 - user/plugin (i.e. uspl2 format of plugin-spec)

Has 29 line(s). Doesn't call other functions.

Uses feature(s): *setopt*

Called by:

.zi-show-report

zi-get-completion-owner

~~~~~

FUNCTION: .zi-get-completion-owner []  
Returns "user---plugin" string (uspl1 format) of plugin that owns given completion.

Both :A and readlink will be used, then readlink's output if results differ.  
Readlink might not be available.

:A will read the link "twice" and give the final repository directory, possibly without username in the uspl format; readlink will read the link "once"

\$1 - absolute path to completion file (in COMPLETIONS\_DIR)  
\$2 - readlink command (":" or "readlink")

Has 20 line(s). Doesn't call other functions.

Uses feature(s): *setopt*

Called by:

.zi-clear-completions  
.zi-get-completion-owner-uspl2col  
.zi-show-completions

*zi-get-completion-owner-uspl2col*

~~~~~

FUNCTION: .zi-get-completion-owner-uspl2col []

For shortening of code - returns colorized plugin name that owns given completion.

\$1 - absolute path to completion file (in COMPLETIONS_DIR)
\$2 - readlink command (":" or "readlink")

Has 2 line(s). Calls functions:

```
.zi-get-completion-owner-uspl2col  
'-- side.zsh/.zi-any-colorify-as-uspl2
```

Called by:

```
.zi-cdisable  
.zi-cenable
```

zi-get-path

~~~~~

FUNCTION: .zi-get-path []  
Returns path of given ID-string, which may be a plugin-spec (like "user/plugin" or "user" "plugin"), an absolute path ("% "/home/..." and also "%SNIPPETS/..." etc.), or a plugin nickname (i.e. id-as'' ice-mod), or a snippet nickname.

Has 5 line(s). Calls functions:

```
.zi-get-path  
'-- zi.zsh/.zi-get-object-path
```

Uses feature(s): *setopt*

Called by:

```
.zi-cd  
.zi-uninstall-completions
```

*zi-glance*

~~~~~

FUNCTION: .zi-glance []
Shows colorized source code of plugin. Is able to use pygmentize,

highlight, GNU source-highlight.

User-action entry point.

\$1 - plugin spec (4 formats: user---plugin, user/plugin, user, plugin)
\$2 - plugin (only when \$1 - i.e. user - given)

Has 37 line(s). Calls functions:

```
.zi-glance  
|-- side.zsh/.zi-exists-physically-message  
|-- side.zsh/.zi-first  
|-- zi.zsh/+zi-message  
'-- zi.zsh/.zi-any-to-user-plugin
```

Not called by script or any function (may be e.g. a hook, a Zle widget, etc.).

zi-help

~~

FUNCTION: .zi-help []
Shows usage information.

User-action entry point.

Has 29 line(s). Calls functions:

```
.zi-help  
'-- zi.zsh/+zi-message
```

Called by:

zi.zsh/zi

zi-list-bindkeys

~~~~

FUNCTION: .zi-list-bindkeys []

Has 39 line(s). Calls functions:

```
.zi-list-bindkeys  
'-- side.zsh/.zi-any-colorify-as-uspl2
```

Called by:

```
zi.zsh/zi
```

### *zi-list-compdef-replay*

~~~~~

```
FUNCTION: .zi-list-compdef-replay []  
Shows recorded compdefs (called by plugins loaded earlier). Plugins often call  
'compdef' hoping  
for 'compinit' being already ran.  ZI  solves this by recording compdefs.
```

```
User-action entry point.
```

Has 5 line(s). Doesn't call other functions.

Called by:

```
zi.zsh/zi
```

zi-ls

~~~

```
FUNCTION: .zi-ls []
```

Has 22 line(s). Doesn't call other functions.

Called by:

```
zi.zsh/zi
```

### *zi-module*

~~~~~

```
FUNCTION: .zi-module []  
Function that has sub-commands passed as long-options (with two dashes, --).
```

It's an attempt to plugin only this one function into 'zi' function defined in zi.zsh, to not make this file longer than it's needed.

Has 33 line(s). Calls functions:

```
.zi-module  
|-- is-at-least  
'-- zi.zsh/+zi-message
```

Uses feature(s): *autoload, is-at-least*

Called by:

```
.zi-build-module  
zi.zsh/Script-Body  
zi.zsh/zi
```

zi-pager

FUNCTION: .zi-pager [][]

Has 27 line(s). Doesn't call other functions.

Uses feature(s): *setopt*

Called by:

```
.zi-glance  
.zi-self-update  
.zi-update-or-status
```

zi-prepare-readlink

FUNCTION: .zi-prepare-readlink [][]
Prepares readlink command, used for establishing completion's owner.

\$REPLY = ":" or "readlink"

Has 4 line(s). Doesn't call other functions.

Uses feature(s): *type*

Called by:

```
.zi-cdisable  
.zi-cenable  
.zi-clear-completions  
.zi-show-completions
```

zi-recall

~~~~~

```
FUNCTION: .zi-recall []
```

Has 34 line(s). Calls functions:

```
.zi-recall  
|-- side.zsh/.zi-compute-ice  
'-- zi.zsh/+zi-deploy-message
```

Uses feature(s): *setopt*

Not called by script or any function (may be e.g. a hook, a Zle widget, etc.).

*zi-recently*

~~~~~

```
FUNCTION: .zi-recently []  
Shows plugins that obtained commits in specified past time.
```

User-action entry point.

\$1 - time spec, e.g. "1 week"

Has 23 line(s). Calls functions:

```
.zi-recently  
'-- side.zsh/.zi-any-colorify-as-uspl2
```

Uses feature(s): *setopt*

Called by:

zi.zsh/zi

zi-registered-ice-mods

~~~~~

FUNCTION: .zi-registered-ice-mods []  
Shows all registerted ice-modifiers.  
Internal and registered by annex.

User-action entry point.

Has 4 line(s). Calls functions:

.zi-registered-ice-mods  
'-- zi.zsh/+zi-message

Called by:

zi.zsh/zi

### *zi-registered-subcommands*

~~~~~

FUNCTION: .zi-registered-subcommands []
Shows subcommands registered by annex.

User-action entry point.

Has 13 line(s). Calls functions:

.zi-registered-subcommands
'-- zi.zsh/+zi-message

Called by:

zi.zsh/zi

zi-restore-extendedglob

~~~~~

```
FUNCTION: .zi-restore-extendedglob []  
Restores extendedglob-option from state saved earlier.
```

Has 1 line(s). Doesn't call other functions.

Uses feature(s): *setopt*

Called by:

```
.zi-format-options  
.zi-unload
```

*zi-run-delete-hooks*

~~~~~

```
FUNCTION: .zi-run-delete-hooks []
```

Has 17 line(s). Calls functions:

```
.zi-run-delete-hooks  
'-- side.zsh/.zi-countdown
```

Uses feature(s): *eval*

Not called by script or any function (may be e.g. a hook, a Zle widget, etc.).

zi-save-set-extendedglob

~~~~~

```
FUNCTION: .zi-save-set-extendedglob []  
Enables extendedglob-option first saving if it was already  
enabled, for restoration of this state later.
```

Has 2 line(s). Doesn't call other functions.

Uses feature(s): *setopt*

Called by:

```
.zi-format-options  
.zi-unload
```

*zi-search-completions*

FUNCTION: `.zi-search-completions` `[[`  
 While `.zi-show-completions()` shows what completions are installed, this functions searches through all plugin dirs showing what's available in general (for installation).

User-action entry point.

Has 39 line(s). Calls functions:

```
.zi-search-completions
'-- side.zsh/.zi-any-colorify-as-uspl2
```

Uses feature(s): *setopt*

Called by:

```
zi.zsh/zi
```

### *zi-self-update*

~~~~~

FUNCTION: `.zi-self-update` `[[`
 Updates `ZI` code (does a git pull).

User-action entry point.

Has 43 line(s). Calls functions:

```
.zi-self-update
|-- zi.zsh/+zi-message
'-- zi.zsh/.zi-get-mtime-into
```

Uses feature(s): *setopt*, *source*, *zcompile*

Called by:

```
.zi-update-or-status-all
zi.zsh/zi
```


zi-show-all-reports

~~~~~

FUNCTION: .zi-show-all-reports []  
Displays reports of all loaded plugins.

User-action entry point.

Has 5 line(s). Calls functions:

.zi-show-all-reports

Called by:

zi.zsh/zi

## zi-show-completions

~~~~~

FUNCTION: .zi-show-completions []
Display installed (enabled and disabled), completions. Detect
stray and improper ones.

Completions live even when plugin isn't loaded - if they are
installed and enabled.

User-action entry point.

Has 62 line(s). Calls functions:

.zi-show-completions
'-- side.zsh/.zi-any-colorify-as-uspl2

Uses feature(s): *setopt*

Called by:

zi.zsh/zi

zi-show-debug-report

FUNCTION: `.zi-show-debug-report` []
Displays dtrace report (data recorded in interactive session).

User-action entry point.

Has 1 line(s). Calls functions:

`.zi-show-debug-report`

Called by:

`zi.zsh/zi`

zi-show-registered-plugins

FUNCTION: `.zi-show-registered-plugins` []
Lists loaded plugins (subcommands list, loaded).

User-action entry point.

Has 19 line(s). Calls functions:

`.zi-show-registered-plugins`
``-- side.zsh/.zi-any-colorify-as-uspl2`

Uses feature(s): *setopt*

Called by:

`zi.zsh/zi`

zi-show-report

FUNCTION: `.zi-show-report` []

Displays report of the plugin given.

User-action entry point.

\$1 - plugin spec (4 formats: user---plugin, user/plugin, user (+ plugin in \$2), plugin)
\$2 - plugin (only when \$1 - i.e. user - given)

Has 66 line(s). Calls functions:

```
.zi-show-report  
|-- zi.zsh/+zi-message  
'-- zi.zsh/.zi-any-to-user-plugin
```

Uses feature(s): *setopt*

Called by:

```
.zi-show-all-reports  
.zi-show-debug-report  
zi.zsh/zi
```

zi-show-times

~~~~~

FUNCTION: *.zi-show-times* [[  
Shows loading times of all loaded plugins.

User-action entry point.

Has 55 line(s). Calls functions:

```
.zi-show-times  
'-- side.zsh/.zi-any-colorify-as-uspl2
```

Uses feature(s): *setopt*

Called by:

```
zi.zsh/zi
```

## zi-show-zstatus

~~~~~

FUNCTION: .zi-show-zstatus []
Shows `z` ZI `z` status, i.e. number of loaded plugins,
of available completions, etc.

User-action entry point.

Has 48 line(s). Calls functions:

```
.zi-show-zstatus  
'-- zi.zsh/+zi-message
```

Uses feature(s): *setopt*

Called by:

```
zi.zsh/zi
```

Environment variables used: `zi.zsh` → `ZPFX`

zi-stress

~~~~~

FUNCTION: .zi-stress []  
Compiles plugin with various options on and off to see how well the code is  
written. The options are:

`NO_SHORT_LOOPS`, `IGNORE_BRACES`, `IGNORE_CLOSE_BRACES`, `SH_GLOB`, `CSH_JUNKIE_QUOTES`,  
`NO_MULTI_FUNC_DEF`.

User-action entry point.

`$1` - plugin spec (4 formats: `user---plugin`, `user/plugin`, `user`, `plugin`)  
`$2` - plugin (only when `$1` - i.e. `user` - given)

Has 31 line(s). Calls functions:

```
.zi-stress
|-- side.zsh/.zi-exists-physically-message
|-- side.zsh/.zi-first
`-- zi.zsh/.zi-any-to-user-plugin
```

Uses feature(s): *setopt*, *zcompile*

Not called by script or any function (may be e.g. a hook, a Zle widget, etc.).

*zi-uncompile-plugin*

~~~~~

FUNCTION: *.zi-uncompile-plugin* [[
Uncompiles given plugin.

User-action entry point.

\$1 - plugin spec (4 formats: user---plugin, user/plugin, user (+ plugin in \$2),
plugin)
\$2 - plugin (only when \$1 - i.e. user - given)

Has 19 line(s). Calls functions:

```
.zi-uncompile-plugin
|-- side.zsh/.zi-any-colorify-as-uspl2
|-- zi.zsh/+zi-message
`-- zi.zsh/.zi-any-to-user-plugin
```

Uses feature(s): *setopt*

Called by:

```
.zi-compile-uncompile-all
zi.zsh/zi
```

zi-uninstall-completions

~~~~~

FUNCTION: *.zi-uninstall-completions* [[  
Removes all completions of given plugin from Zshell (i.e. from FPATH).  
The FPATH is typically '~/.zi/completions/'.

```
$1 - plugin spec (4 formats: user---plugin, user/plugin, user, plugin)
$2 - plugin (only when $1 - i.e. user - given)
```

Has 39 line(s). Calls functions:

```
.zi-uninstall-completions
|-- install.zsh/.zi-compinit
|-- install.zsh/.zi-forget-completion
`-- zi.zsh/+zi-message
```

Uses feature(s): *setopt*, *source*

Called by:

```
zi.zsh/zi
```

*zi-unload*

~~~~~

```
FUNCTION: .zi-unload [[[
0. Call the Zsh Plugin's Standard *_plugin_unload function
0. Call the code provided by the Zsh Plugin's Standard @zsh-plugin-run-at-update
1. Delete bindkeys (...)
2. Delete Zstyles
3. Restore options
4. Remove aliases
5. Restore Zle state
6. Unfunction functions (created by plugin)
7. Clean-up FPATH and PATH
8. Delete created variables
9. Forget the plugin
```

User-action entry point.

```
$1 - plugin spec (4 formats: user---plugin, user/plugin, user, plugin)
$2 - plugin (only when $1 - i.e. user - given)
```

Has 380 line(s). Calls functions:

```
.zi-unload
|-- additional.zsh/.zi-clear-debug-report
|-- side.zsh/.zi-any-colorify-as-uspl2
```

```
|-- zi.zsh/+zi-message
`-- zi.zsh/.zi-any-to-user-plugin
```

Uses feature(s): *alias, bindkey, eval, setopt, unalias, unfunction, zle, zstyle*

Called by:

```
additional.zsh/.zi-debug-unload
zi.zsh/.zi-run-task
zi.zsh/zi
```

zi-unregister-plugin

~~~~~

```
FUNCTION: .zi-unregister-plugin []
Removes the plugin from ZI_REGISTERED_PLUGINS array and from the
zsh_loaded_plugins array (managed according to the plugin standard)
```

Has 5 line(s). Calls functions:

```
.zi-unregister-plugin
`-- zi.zsh/.zi-any-to-user-plugin
```

Called by:

```
.zi-unload
```

### *zi-update-all-parallel*

~~~~~

```
FUNCTION: .zi-update-in-parallel []
```

Has 63 line(s). Calls functions:

```
.zi-update-all-parallel
|-- side.zsh/.zi-any-colorify-as-uspl2
|-- zi.zsh/+zi-message
`-- zi.zsh/.zi-any-to-user-plugin
```

Uses feature(s): *setopt*

Called by:

```
.zi-update-or-status-all
```

zi-update-or-status

~~~~~

FUNCTION: `.zi-update-or-status` `[[`  
Updates (git pull) or does 'git status' for given plugin.

User-action entry point.

\$1 - "status" for status, other for update  
\$2 - plugin spec (4 formats: user---plugin, user/plugin, user (+ plugin in \$2), plugin)  
\$3 - plugin (only when \$1 - i.e. user - given)

Has 280 line(s). Calls functions:

```
.zi-update-or-status
|-- install.zsh/.zi-get-latest-gh-r-url-part
|-- install.zsh/.zi-setup-plugin-dir
|-- side.zsh/.zi-any-colorify-as-uspl2
|-- side.zsh/.zi-compute-ice
|-- side.zsh/.zi-exists-physically
|-- side.zsh/.zi-exists-physically-message
|-- side.zsh/.zi-store-ices
|-- side.zsh/.zi-two-paths
|-- zi.zsh/+zi-message
|-- zi.zsh/.zi-any-to-user-plugin
`-- zi.zsh/.zi-set-m-func
```

Uses feature(s): *kill, read, setopt, source, trap, wait*

Called by:

```
.zi-update-all-parallel
.zi-update-or-status-all
zi.zsh/zi
```

*zi-update-or-status-all*

~~~~~

FUNCTION: `.zi-update-or-status-all` `[[`

Updates (git pull) or does 'git status' for all existing plugins. This includes also plugins that are not loaded into Zsh (but exist on disk). Also updates (i.e. redownloads) snippets.

User-action entry point.

Has 103 line(s). Calls functions:

```
.zi-update-or-status-all
|-- install.zsh/.zi-compinit
|-- side.zsh/.zi-any-colorify-as-uspl2
|-- zi.zsh/+zi-message
|-- zi.zsh/.zi-any-to-user-plugin
`-- zi.zsh/.zi-get-mtime-into
```

Uses feature(s): *setopt*, *source*

Called by:

zi.zsh/zi

zi-update-or-status-snippet

~~~~~<sup>~</sup>

FUNCTION: .zi-update-or-status-snippet []

Implements update or status operation for snippet given by URL.

\$1 - "status" or "update"  
\$2 - snippet URL

Has 28 line(s). Calls functions:

```
.zi-update-or-status-snippet
|-- install.zsh/.zi-update-snippet
`-- side.zsh/.zi-compute-ice
```

Uses feature(s): *source*

Called by:

```
.zi-update-all-parallel  
.zi-update-or-status-all  
.zi-update-or-status
```

*zi-wait-for-update-jobs*

```
]]]  
FUNCTION: .zi-wait-for-update-jobs [[[
```

Has 14 line(s). Calls functions:

```
.zi-wait-for-update-jobs  
'-- zi.zsh/+zi-message
```

Uses feature(s): *wait*

Called by:

```
.zi-update-all-parallel
```

## is-at-least

Test whether \$ZSH\_VERSION (or some value of your choice, if a second argument is provided) is greater than or equal to x.y.z-r (in argument one). In fact, it'll accept any dot/dash-separated string of numbers as its second argument and compare it to the dot/dash-separated first argument. Leading non-number parts of a segment (such as the "zefram" in 3.1.2-zefram4) are not considered when the comparison is done; only the numbers matter. Any left-out segments in the first argument that are present in the version string compared are considered as zeroes, eg 3 == 3.0 == 3.0.0 == 3.0.0.0 and so on.

Has 56 line(s). Doesn't call other functions.

Called by:

```
.zi-module
```