This page is a description of the current architecture and design of Oh My Zsh.

This is a work in progress: we don't think there's anything outright wrong in it, but much may still be left out.

This page is not authoritative or normative: it was put together by Oh My Zsh users based on the current Oh My Zsh code; it's not a design document from the original Oh My Zsh authors. But it should serve as a useful guide to users and developers who want to get familiar with the Oh My Zsh codebase.

Overview

What Oh My Zsh provides:

- Configuration of zsh itself, enabling advanced features
- Theming
- Extra shell functionality & terse shortcuts
  - e.g. dir navigation
- Integration with third party programs and commands
  - Completion, aliases, and auxiliary functions
  - Both OS-specific stuff and applications/utilities
- Auto-starting programs
  - e.g. the gpg-agent plugin

It seems that plugins can get arbitrarily powerful and do whatever they want, so the user should understand what a given plugin does before enabling it.

Variables

These are variables that base OMZ (excluding any plugins) uses. I've read through .oh-my-zsh so far, but not the lib/*.zsh files. More may be on the way.

Variables OMZ reads

At initialization time:

In oh-my-zsh.sh:

- `ZSH` - path to .oh-my-zsh (not zsh) installation
- `plugins` - user-provided list of plugins to load
- `ZSH_CUSTOM` – path to the Oh My Zsh (not zsh itself) customization dir
- `ZSH_THEME` – theme to load at startup
- `CASE_SENSITIVE` – controls zsh completion matching
- `COMPLETION_WAITING_DOTS`
- `DISABLE_AUTO_UPDATE` – ("true"/*)
- `DISABLE_AUTO_PROMPT` – ("true"/*)

- `DISABLE_LS_COLORS` – in lib/theme-and-appearance
- `ENABLE_CORRECTION`
- `ZSH_CACHE_DIR`
- `ZSH_COMPDUMP`
- `ZSH_VERSION`
- `ZDOTDIR`
- Standard Unix environment variables
  - `HOME`
  - `HOST`
  - `OSTYPE`

The only required one is `$ZSH`. The rest either have defaults, or undef is fine and has an expected behavior. (Though if $plugins is unset, the git plugin won't get loaded. Will that break stuff?) All these `ZSH_*` variables are OMZ-specific, not standard `zsh` stuff. Except for `ZSH_VERSION`, which is a standard parameter provided by `zsh`.

Variables OMZ provides or modifies

In oh-my-zsh.sh:

At init:

- `SHORT_HOST`
- `LSCOLORS`
- `SCREEN_NO`
- `PS1`
- `POST_1_7_2_GIT`
- `PAGER`
- `LESS`
- `FX` – special terminal control "effects" (reset/bold/no-bold/etc)
- `FG`
- `BG`

At init (defaults if not provided):

- `ZSH_CUSTOM` - defaults to `$ZSH/custom`
- `ZSH_CACHE_DIR` - defaults to `$ZSH/cache`
- `ZSH_COMPDUMP`
- `ZSH_SPECTRUM_TEXT`

Modified at init:

- `fpath`
- `LC_CTYPE`

Leaks:

- `custom_config_file`

Variables plugins use

- `URLTOOLS_METHOD` - plugins/urltools uses it to manually select node/php/perl/python/etc

Oh My Zsh Initialization

Oh My Zsh is initialized for the current `zsh` session by sourcing `$ZSH/oh-my-zsh.sh`. This is typically done from `.zshrc`, and the Oh My Zsh installation process modifies the user's `.zshrc` to do so.

The basic steps of the Oh My Zsh initialization process are as follows. Note that the order of steps is subject to change.

- Check for updates
- Path defaulting
- Load libs
- Load custom user code
- Initialize completion system
- Load plugins
- Load theme

The initialization steps in detail:

- Check for updates
  - Runs in a separate `zsh` process
  - Does not load OMZ, so it's independent and doesn't use any OMZ files
- Update `$fpath`: Add functions/ and completions/
  - (even though they don't exist in the current codebase)
- Set `$ZSH_CUSTOM` and `$ZSH_CACHE_DIR`
- Load lib ("config") files
  - Discovers and sources all lib files, in alphabetical order, respecting custom overrides
- Load custom user code
  - Source each `$ZSH_CUSTOM/*.zsh` file, in alphabetical order
- Pre-load plugins (add to `$fpath`, but don't source)
- Set `$SHORT_HOST`
- Initialize Completion support
  - Set `$ZSH_COMPDUMP`
  - Run `compinit`, using dump file
- Load plugins
  - Source each plugin specified in `$plugins`, in the order specified
- Load theme

Customization

In Oh My Zsh terms, *customization* means adding or overriding zsh code, including its internals and implementation. It's not just a term for user-specified configuration.

Overriding internals can be done by adding `*.zsh` files to the `$ZSH_CUSTOM` root directory. All `*.zsh` files there will be sourced after OMZ loads and sources its own lib/* files. This allows you to redefine functions after the fact. (This will take place after any setup has called OMZ functions.) These are referred to as "config files" in oh-my-zsh.sh.

It's not documented in the *Customization* page, but `$ZSH_CUSTOM/lib/*.zsh` do override the corresponding internals lib files. If a custom one is present, it is sourced instead of the one in the distribution.

So, you can:

- Override lib/* files on a per-file basis (loaded instead of the base file of the same name)
- Add arbitrary customization code that runs later and can redefine any function or variable from the core
- Override plugins and themes on a per-plugin/theme basis (loaded instead of base)
- Override parts of plugins by defining an additional "patch" plugin and including it in `$plugins` *after* the base plugin

`$ZSH_CUSTOM` controls where the custom override files are found; defaults to `$ZSH/custom` (under the main OMZ installation).

As of June 2015, user "custom" files are loaded before plugins are loaded, so they cannot be used to modify plugins. To customize a plugin by replacing selected functions or variables, you need to define an additional custom plugin and load that after the base plugin you wish to customize.

The Customization wiki page doesn't discuss which functions/APIs are stable. It mostly talks about overriding on a per-file basis.

Plugins

Oh My Zsh plugins extend the core functionality of Oh My Zsh.

Plugins provide functionality in the following areas:

- Completion definitions
- Functions
- Aliases

A "completion plugin" is the term for a plugin that has nothing but completion system definitions. They are not handled or loaded differently from other plugins.

Plugins are optional, and selected at runtime. When Oh My Zsh is initialized, only the plugins specified in the user-defined `$plugins` variable are loaded. The core Oh My Zsh code does not depend on any plugins. Themes may depend on plugins. There's no standard mechanism to express these dependencies, though themes should note their plugin dependencies in their comments.

The plugins live in `plugins/` in the Oh My Zsh source tree. Even though their source code is in the main Oh My Zsh repository, each plugin has its own maintainer. The maintainers are listed on the Plugins page, or in the source code of the plugin.

Themes

Themes control the appearance of the `zsh` prompt, the appearance of certain other programs, and some other behaviors, such as tab and window titles in terminal emulators.

OMZ turns on the `prompt_subst` shell option, and OMZ themes assume it is enabled.

Themes set a variety of variables to control the appearance of the zsh prompt. They may also install hook functions. These variables are read by core OMZ functions like `git_prompt_info()` and used to modify their behavior and style their output.

Things themes do:

- Set `$PROMPT`, `$RPROMPT`, and related variables
- Set `$LSCOLORS` and `$LS_COLORS`
- Define hook functions

Variables used by themes

These variables are set by themes to control the prompt's appearance and other cosmetic behavior.

- `PROMPT`
- `DEFAULT_USER`
- `ZSH_THEME_SCM_PROMPT_PREFIX` – used in `bzr_prompt_info()` from `lib/bzr.sh`

git_prompt_info():

- `ZSH_THEME_GIT_PROMPT_PREFIX`
- `ZSH_THEME_GIT_PROMPT_SUFFIX`
- `ZSH_THEME_GIT_COMMITS_AHEAD_PREFIX`
- `ZSH_THEME_GIT_COMMITS_AHEAD_SUFFIX`
- `ZSH_THEME_GIT_PROMPT_DIRTY`
- `ZSH_THEME_GIT_PROMPT_CLEAN`
- `ZSH_THEME_GIT_PROMPT_BEHIND_REMOTE`

- `ZSH_THEME_GIT_PROMPT_AHEAD_REMOTE`
- `ZSH_THEME_GIT_PROMPT_DIVERGED_REMOTE`
- `ZSH_THEME_GIT_PROMPT_AHEAD`
- `ZSH_THEME_GIT_PROMPT_BEHIND`
- `ZSH_THEME_GIT_PROMPT_SHA_BEFORE`
- `ZSH_THEME_GIT_PROMPT_SHA_AFTER`
- `ZSH_THEME_GIT_PROMPT_ADDED`
- `ZSH_THEME_GIT_PROMPT_MODIFIED`
- `ZSH_THEME_GIT_PROMPT_RENAMED`
- `ZSH_THEME_GIT_PROMPT_DELETED`
- `ZSH_THEME_GIT_PROMPT_STASHED`
- `ZSH_THEME_GIT_PROMPT_UNMERGED`
- `ZSH_THEME_GIT_PROMPT_DIVERGED`
- `ZSH_THEME_GIT_PROMPT_UNTRACKED`

nvm_prompt_info():

- `ZSH_THEME_NVM_PROMPT_PREFIX`
- `ZSH_THEME_NVM_PROMPT_SUFFIX`

rvm_prompt_info():

- `ZSH_THEME_RVM_PROMPT_PREFIX`
- `ZSH_THEME_RVM_PROMPT_SUFFIX`
- `ZSH_THEME_RVM_PROMPT_OPTIONS`

lib/termsupport.zsh:

- `ZSH_THEME_TERM_TAB_TITLE_IDLE`

- `ZSH_THEME_TERM_TITLE_IDLE`

- `chpwd_functions`

- `precmd_functions`

Not all themes use all of these variables. Most use only a subset.

Other `ZSH_THEME_*` variables may be used by different themes and plugins. The pattern seems to be that they will supply a `XXX_prompt_info()` function which can be configured using those variables.

Some per-theme custom variables:

In bureau.zsh-theme:

- `ZSH_THEME_GIT_PROMPT_STAGED`
- `ZSH_THEME_GIT_PROMPT_UNSTAGED`
- `ZSH_THEME_GIT_PROMPT_UNTRACKED`

These variables are mostly used in prompt info functions. To use them to customize a theme's prompt, the theme should set the various `ZSH_THEME_*` variables to contain text that will get interpolated in at the appropriate spots.

And then put one or more of these output-capturing function calls inside the `PROMPT` variable, if you are defining a custom `PROMPT`:

- $(git_prompt_info)
- $(bzr_prompt_info)
- $(nvm_prompt_info)
- $(rvm_prompt_info)
- $(ruby_prompt_info)

Or use other `*_prompt_info` functions that plugns define. These `prompt_info` functions have dummy implementations (in `lib/prompt_info_functions.zsh`) so they can be used unconditionally in theme prompts and will gracefully degrade to outputting empty strings if the appropriate plugin is not actually loaded.

The Oh My Zsh prompt construction functions (found inside `lib/`) send their output to `stdout`, and are designed to be called with the `$(...)` output-capturing subshell invocation.

Themes use color definitions from zsh's color definitions. (`autoload -U colors && colors`).

Some themes may define additional functions, and set `precmd` or `chpwd` hooks. There's no defined mechanism for cleaning these up. It looks like themes are assumed to be set up once during initialization, and then not turned off or changed during that session.

Although some existing themes set `$chpwd` or `$precmd`, it's probably better for themes to add functions to the `$chpwd_functions` or `$precmd_functions` lists, to avoid conflicting with other code that may also want to set hooks.

Loading themes

The Oh My Zsh theme mechanism is designed to load a theme once per session, during OMZ initialization.

The theme mechanism does not provide a way to unload themes. The values for `PROMPT`, `RPROMPT`, `ZSH_THEME_*`, and hooks do not get reset. Thus, you can hack in support for switching themes during a session, but it is not clean: when you switch themes, you can get leftover settings from previously loaded themes and end up with a combination of themes.

Miscellaneous Architecture Information

Plugins or other files that need to locate where they're running from should use `${0:h}` instead of a path relative to `$ZSH`, so they properly find themselves when running from the `$ZSH_CUSTOM` directory or if their plugin name changes.