Zsh Scripting Cheatsheet

A handy reference for Zsh scripting, covering syntax, built-ins, options, and common patterns.

Zsh Basics & Syntax

Basic Syntax

Comments	`# This is a comment`
Variables	`VAR_NAME="value"` `echo $VAR_NAME` or `echo ${VAR_NAME}`
String Concatenation	`result="Hello, "$VAR_NAME` `result="Hello, ${VAR_NAME}"`
Command Substitution	`output=$(ls -l)` or `output=` ls -l
Exit Status	`$?` (Exit status of the last command)
Here Documents	`cat <<EOF Multiline text EOF`
Arrays	`my_array=(item1 item2 item3)` `echo ${my_array[0]}` `echo ${#my_array[@]}` (length) `echo ${my_array[@]}` (all elements)`

Conditional Statements

```
if [[ condition ]]; then
  # code
elif [[ condition ]]; then
  # code
else
  # code
fi
```

String Comparisons:

`==` , `!=` , `<` , `>`

File Tests:

`-e file` (exists), `-f file` (regular file), `-d file` (directory), `-r file` (readable), `-w file` (writable), `-x file` (executable)

Numeric Comparisons:

`-eq` , `-ne` , `-lt` , `-gt` , `-le` , `-ge`

Looping

For Loop:

```
for var in item1 item2 item3; do
  echo $var
done
```

While Loop:

```
while [[ condition ]]; do
  # code
done
```

Until Loop:

```
until [[ condition ]]; do
  # code
done
```

Looping through array:

```
for element in "${my_array[@]}"; do
  echo "Element: $element"
done
```

Functions & Builtins

Functions

Function Definition	`my_function() { echo "Hello from my_function" }`
	OR
	`function my_function { echo "Hello from my_function" }`

Calling a function	`my_function`
Passing Arguments	`my_function arg1 arg2` Inside function: `$1` , `$2` , etc.
Return Value	`return 0` (Success), `return 1` (Failure). Use `$?` to check.
Local Variables	`local my_var="value"` (Scope is within the function)

Useful Built-in Commands

`echo`	Print text to standard output.
`printf`	Formatted output (similar to C's `printf`).
`read`	Read input from standard input. `read -p "Prompt: " variable`
`exit`	Exit the script. `exit 0` (Success), `exit 1` (Failure).
`source` or `.`	Execute commands from a file in the current shell. `. ./my_script.sh`
`pwd`	Print working directory.
`cd`	Change directory.

Parameter Expansion

${var:-word} - Use default value 'word' if var is unset or null.

${var:=word} - Assign default value 'word' to var if var is unset or null.

${var:?message} - Display error message and exit if var is unset or null.

${var:+word} - Use 'word' if var is set and not null.

${#var} - String length of var.

${var:offset:length} - Substring of var.

Zsh Options & Globbing

Setting Options

`set -o option_name`	Enable option.
`set +o option_name`	Disable option.
`setopt option_name`	Enable option (Zsh specific).
`unsetopt option_name`	Disable option (Zsh specific).

Useful Options

allexport: Automatically export all defined variables.

errexit: Exit immediately if a command exits with a non-zero status.

nounset: Treat unset variables as an error when substituting.

xtrace: Print commands and their arguments as they are executed.

noclobber: Prevent overwriting existing files with redirection.

Globbing (Filename Generation)

`*`	Matches any string of characters (except leading dots).
`?`	Matches any single character.
`[abc]`	Matches one of the characters `a`, `b`, or `c`.
`[a-z]`	Matches any character between `a` and `z`.
`[^abc]` or `[!abc]`	Matches any character except `a`, `b`, or `c`.
`(foo|bar)`	Matches either `foo` or `bar`.
`**`	Matches files recursively (Zsh specific, enable with `setopt glob_star`).

Advanced Zsh

Zsh Modules

zmodload zsh/module	Loads a specific Zsh module. Example: `zmodload zsh/tcp`
Available Modules	`zmodload -l` Lists available modules.
Common modules	zsh/tcp, zsh/datetime, zsh/mathfunc

Process Substitution

<(command)	Provides the output of command as a file. `diff <(ls dir1) <(ls dir2)`
>(command)	Redirects standard output to command. `tee >(gzip > file.gz) > file.txt`

Customizing the Prompt

PS1 is the primary prompt variable.

`PS1="%n@%m %~> "` (user@host current_directory>)

Common escape sequences: %n (username), %m (hostname), %~ (current directory), %d (current directory full path), %w (current time), %D{format} (date/time with format).

Example:

```
PS1='%F{blue}%n@%m%f:%F{green}%~%f$(git_prompt_info)'
```

For advanced prompt customization, use a Zsh theme manager like Oh My Zsh or Prezto.