fish shell Friendly interactive shell

START HERE

`Right`
Accept the grey suggestion

`function <name> ... end`
Define a function

`Tab`
Complete and open the pager

`funcsave <name>`
Keep a function for good

`abbr -a <name> <text>`
Short name that expands

`fish_add_path <dir>`
Add a directory to PATH

`set -gx <NAME> <value>`
Export a variable

`fish_config`
Pick prompt and theme

CONCEPTS

fish suggests, highlights and completes out of the box. Most setup is a few lines in config.fish, not a plugin framework.

Universal `-U` all sessions, saved
Global `-g` this shell
Function `-f` inside a function
Block `-l` inside if/for/begin

set scope, widest to narrowest

fish is not POSIX: no word splitting, no $VAR=x, no heredocs. Run bash scripts with bash, not by sourcing them.

Single quotes expand nothing; double quotes expand $var and $(cmd). Unquoted lists expand to one argument per element.

SUGGESTIONS & COMPLETION

`Right / Ctrl+F`
Accept the whole autosuggestion

`Alt+Right / Alt+F`
Accept one word of the suggestion

`Tab`
Complete; with several matches open the pager

`~/.config/fish/completions/<cmd>.fish`
Where your own completions live

COMMAND-LINE KEYS

`Ctrl+A / Ctrl+E`
Start / end of line

`Ctrl+W`
Delete the previous path component

`Ctrl+U`
Delete to start of line

`Ctrl+L`
Clear the screen, keep the line

`Ctrl+C`
Cancel the current line

`Ctrl+D`
Delete char; exit on an empty line

`Alt+E`
Edit the line in $EDITOR

`Alt+S`
Prepend sudo to the line

COMMAND-LINE KEYS (cont.)

`Alt+H`
Show the man page for this command

`fish_vi_key_bindings`
Switch to vi key bindings
Go back with fish_default_key_bindings. Esc leaves insert mode.

`fish_key_reader`
Show the name of a pressed key

HISTORY & DIRECTORIES

`Up / Down`
Search history for what you typed

`Ctrl+R`
Open history in the pager

`Alt+Up / Alt+Down`
Search history for the token under cursor

`history search <text>`
List matching history entries

`fish --private`
Session that does not save history

VARIABLES & LISTS

`set <name> <value>`
Set a variable

`set -gx <NAME> <value>`
Set global and export

`set -U <name> <value>`
Universal: kept across sessions

`set -l <name> <value>`
Local to the current block

`set -e <name>`
Erase a variable

`set -q <name>`
Succeeds if the variable is set

`fish_add_path <dir>`
Add a directory to PATH (saved)

`set <name> a b c`
A variable is a list

`$name[2]`
Element 2 (lists start at 1)

`$status`
Exit status of the last command

`$argv`
Arguments of the function or script

EXPANSION

`(<cmd>)`
Command substitution, one arg per line

EXPANSION (cont.)

`**/*.txt`
Recurse into subdirectories

CONTROL FLOW

Conditions

```
if test -f <file>
    echo yes
else if test -d <file>
    echo dir
else
    echo no
end
```

Loops

```
for f in *.txt
    echo $f
end

while read -l line
    echo $line
end < <file>
```

`test -e <path>`
True if the path exists

`<a>; and <b>`
Run b only if a succeeds
&& and || work too. not <cmd> negates a command.

`math <a> + <b>`
Arithmetic, floats included

FUNCTIONS & ABBREVIATIONS

`function <name> ... end`
Define a function ($argv holds args)

`funcsave <name>`
Save a function to ~/.config/fish/functions

`abbr -a <name> <text>`
Expand name to text on space or Enter

`Ctrl+Space`
Space without expanding the abbreviation

STRINGS

`string match -r '<regex>' <text>`
Regex match, prints the match

`string replace -a <old> <new> <text>`
Replace all occurrences

`string split <sep> <text>`
Split into list items

REDIRECTION & PIPES

`<cmd> > <file>`
Stdout to a file

`<cmd> >> <file>`
Append stdout

`<cmd> 2> <file>`
Stderr to a file

REDIRECTION & PIPES (cont.)

`<cmd> &> <file>`
Stdout and stderr to a file

fish has no heredoc: pipe from printf or use a quoted multi-line string.

CONFIG & PROMPT

`~/.config/fish/config.fish`
Runs at every start

`~/.config/fish/functions/<name>.fish`
Autoloaded one function per file

`fish_config`
Web UI for prompt and theme

COMING FROM BASH

bash to fish

bash	fish
`export X=1`	`set -gx X 1`
`$?`	`$status`
`$(cmd)`	`(cmd) or $(cmd)`
`${v/a/b}`	`string replace a b $v`
`$((1+2))`	`math 1 + 2`
`<(cmd)`	`(cmd | psub)`
`[[ $a == b ]]`	`test $a = b`
`f() { ...; }`	`function f ... end`
`cmd1 && cmd2`	`cmd1; and cmd2`

Unquoted $var never splits on spaces and a glob with no match is an error, not passed through literally.

RECIPES

Starter config.fish

```
# ~/.config/fish/config.fish
fish_add_path ~/.local/bin
set -gx EDITOR vim
abbr -a gco git checkout
abbr -a gs git status
set -g fish_greeting
```

Function with options

```
function greet -d 'Say hello'
    argparse 'n/name=' -- $argv
    echo Hello $_flag_name
end
funcsave greet
```

Rename in bulk

```
for f in *.jpeg
    mv $f (string
replace .jpeg .jpg $f)
end
```

Capture multi-line output

```
set -l out (git status --short |
string collect)
echo "$out"
# without collect, $out would be
a list of lines
```

HELP

`help <topic>`
Open the docs on a topic