Help

August 25, 2026

Help

Kind: Class

Source: `lib/help.js`

TypeScript import types for JSDoc, used by Visual Studio Code IntelliSense and `npm run typescript-checkJS`
https://www.typescriptlang.org/docs/handbook/jsdoc-supported-types.html#import-types

`Help` prepares command metadata for help output, including visible commands, options, global options, and arguments. It also creates display terms and comparison data so command help can be ordered and aligned consistently.

Methods

Method	Signature	Returns
`prepareContext`	`prepareContext(contextOptions: undefined)`	`void`
`visibleCommands`	`visibleCommands(cmd: undefined)`	`void`
`compareOptions`	`compareOptions(a: undefined, b: undefined)`	`void`
`visibleOptions`	`visibleOptions(cmd: undefined)`	`void`
`visibleGlobalOptions`	`visibleGlobalOptions(cmd: undefined)`	`void`
`visibleArguments`	`visibleArguments(cmd: undefined)`	`void`
`subcommandTerm`	`subcommandTerm(cmd: undefined)`	`void`
`optionTerm`	`optionTerm(option: undefined)`	`void`
`argumentTerm`	`argumentTerm(argument: undefined)`	`void`
`longestSubcommandTermLength`	`longestSubcommandTermLength(cmd: undefined, helper: undefined)`	`void`

Method	Signature	Returns
`longestOptionTermLength`	`longestOptionTermLength(cmd: undefined, helper: undefined)`	`void`
`longestGlobalOptionTermLength`	`longestGlobalOptionTermLength(cmd: undefined, helper: undefined)`	`void`
`longestArgumentTermLength`	`longestArgumentTermLength(cmd: undefined, helper: undefined)`	`void`
`commandUsage`	`commandUsage(cmd: undefined)`	`void`
`commandDescription`	`commandDescription(cmd: undefined)`	`void`
`subcommandDescription`	`subcommandDescription(cmd: undefined)`	`void`
`optionDescription`	`optionDescription(option: undefined)`	`void`
`argumentDescription`	`argumentDescription(argument: undefined)`	`void`
`formatItemList`	`formatItemList(heading: undefined, items: undefined, helper: undefined)`	`void`
`groupItems`	`groupItems(unsortedItems: undefined, visibleItems: undefined, getGroup: undefined)`	`void`
`formatHelp`	`formatHelp(cmd: undefined, helper: undefined)`	`void`
`displayWidth`	`displayWidth(str: undefined)`	`void`
`styleTitle`	`styleTitle(str: undefined)`	`void`
`styleUsage`	`styleUsage(str: undefined)`	`void`
`styleCommandDescription`	`styleCommandDescription(str: undefined)`	`void`
`styleOptionDescription`	`styleOptionDescription(str: undefined)`	`void`
`styleSubcommandDescription`	`styleSubcommandDescription(str: undefined)`	`void`
`styleArgumentDescription`	`styleArgumentDescription(str: undefined)`	`void`
`styleDescriptionText`	`styleDescriptionText(str: undefined)`	`void`
`styleOptionTerm`	`styleOptionTerm(str: undefined)`	`void`
`styleSubcommandTerm`	`styleSubcommandTerm(str: undefined)`	`void`
`styleArgumentTerm`	`styleArgumentTerm(str: undefined)`	`void`
`styleOptionText`	`styleOptionText(str: undefined)`	`void`

Method	Signature	Returns
`styleArgumentText`	`styleArgumentText(str: undefined)`	`void`
`styleSubcommandText`	`styleSubcommandText(str: undefined)`	`void`
`styleCommandText`	`styleCommandText(str: undefined)`	`void`
`padWidth`	`padWidth(cmd: undefined, helper: undefined)`	`void`
`preformatted`	`preformatted(str: undefined)`	`void`
`formatItem`	`formatItem(term: undefined, termWidth: undefined, description: undefined, helper: undefined)`	`void`
`boxWrap`	`boxWrap(str: undefined, width: undefined)`	`void`

Where it refuses work

- `Help` stops the work with an early return when `word === '[options]'`, in 2 places.
- `Help` stops the work with an early return when `word[0] === '[' || word[0] === '<'`, in 2 places.
- `Help` stops the work with an early return when `!this.showGlobalOptions`.
- `Help` stops the work with an early return when `cmd.registeredArguments.find((argument) => argument.description)`.
- `Help` stops the work with an early return when `option.description`.
- `Help` stops the work with an early return when `argument.description`.

Diagram

```
graph LR
  Command[Command definition] --> Help[Help]
  Help --> Context[Prepared context]
  Context --> Commands[Visible commands]
  Context --> Options[Visible options]
  Context --> Arguments[Visible arguments]
  Options --> Terms[Option terms]
  Commands --> Terms
  Arguments --> Terms
  Terms --> HelpOutput[Help output]
```

Usage

```
import { Command, Help } from 'commander';

const program = new Command()
  .name('deploy')
  .description('Deploy an application')
  .argument('<environment>', 'target environment')
  .option('-d, --dry-run', 'show changes without deploying');

const help = new Help();

help.prepareContext();

const optionTerms = help
  .visibleOptions(program)
  .map((option) => help.optionTerm(option));

const argumentTerms = help
  .visibleArguments(program)
  .map((argument) => help.argumentTerm(argument));

console.log({
  options: optionTerms,
  arguments: argumentTerms,
});
```

AI Coding Instructions

- Call `prepareContext()` before reading visible commands, options, or arguments when generating help content.
- Use `visibleOptions()`, `visibleGlobalOptions()`, and `visibleCommands()` instead of reading command collections directly so hidden entries are excluded.
- Build displayed labels through `subcommandTerm()`, `optionTerm()`, and `argumentTerm()` rather than duplicating formatting logic.
- Use `compareOptions()` when sorting option lists to keep help output ordering consistent.
- Use `longestSubcommandTermLength()` when calculating padding for aligned subcommand output.

Relationships

- IMPORTS → `humanReadableArgName`

Used by

2 references from 2 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (2)

- `program` — `index.js`:7
- `Command` — `lib/command.js`:14