Command

August 25, 2026

Command

Kind: Class

Source: `lib/command.js`

`Command` represents a CLI command and manages its child-command hierarchy, inherited settings, help output, and error behavior. It creates commands and help instances, applies output configuration, and controls help or suggestion display after errors.

Extends: `EventEmitter`

Methods

Method	Signature	Returns
`copyInheritedSettings`	`copyInheritedSettings(sourceCommand: undefined)`	`void`
`_getCommandAndAncestors`	`_getCommandAndAncestors()`	`void`
`command`	`command(nameAndArgs: undefined, actionOptsOrExecDesc: undefined, execOpts: undefined)`	`void`
`createCommand`	`createCommand(name: undefined)`	`void`
`createHelp`	`createHelp()`	`void`
`configureHelp`	`configureHelp(configuration: undefined)`	`void`
`configureOutput`	`configureOutput(configuration: undefined)`	`void`
`showHelpAfterError`	`showHelpAfterError(displayHelp: undefined)`	`void`
`showSuggestionAfterError`	`showSuggestionAfterError(displaySuggestion: undefined)`	`void`

Method	Signature	Returns
`addCommand`	`addCommand(cmd: undefined, opts: undefined)`	`void`
`createArgument`	`createArgument(name: undefined, description: undefined)`	`void`
`argument`	`argument(name: undefined, description: undefined, parseArg: undefined, defaultValue: undefined)`	`void`
`arguments`	`arguments(names: undefined)`	`void`
`addArgument`	`addArgument(argument: undefined)`	`void`
`helpCommand`	`helpCommand(enableOrNameAndArgs: undefined, description: undefined)`	`void`
`addHelpCommand`	`addHelpCommand(helpCommand: undefined, deprecatedDescription: undefined)`	`void`
`_getHelpCommand`	`_getHelpCommand()`	`void`
`hook`	`hook(event: undefined, listener: undefined)`	`void`
`exitOverride`	`exitOverride(fn: undefined)`	`void`
`_exit`	`_exit(exitCode: undefined, code: undefined, message: undefined)`	`void`
`action`	`action(fn: undefined)`	`void`
`createOption`	`createOption(flags: undefined, description: undefined)`	`void`
`_callParseArg`	`_callParseArg(target: undefined, value: undefined, previous: undefined, invalidArgumentMessage: undefined)`	`void`
`_registerOption`	`_registerOption(option: undefined)`	`void`
`_registerCommand`	`_registerCommand(command: undefined)`	`void`
`addOption`	`addOption(option: undefined)`	`void`
`_optionEx`	`_optionEx(config: undefined, flags: undefined, description: undefined, fn: undefined, defaultValue: undefined)`	`void`

Method	Signature	Returns
`option`	`option(flags: undefined, description: undefined, parseArg: undefined, defaultValue: undefined)`	`void`
`requiredOption`	`requiredOption(flags: undefined, description: undefined, parseArg: undefined, defaultValue: undefined)`	`void`
`combineFlagAndOptionalValue`	`combineFlagAndOptionalValue(combine: undefined)`	`void`
`allowUnknownOption`	`allowUnknownOption(allowUnknown: undefined)`	`void`
`allowExcessArguments`	`allowExcessArguments(allowExcess: undefined)`	`void`
`enablePositionalOptions`	`enablePositionalOptions(positional: undefined)`	`void`
`passThroughOptions`	`passThroughOptions(passThrough: undefined)`	`void`
`_checkForBrokenPassThrough`	`_checkForBrokenPassThrough()`	`void`
`storeOptionsAsProperties`	`storeOptionsAsProperties(storeAsProperties: undefined)`	`void`
`getOptionValue`	`getOptionValue(key: undefined)`	`void`
`setOptionValue`	`setOptionValue(key: undefined, value: undefined)`	`void`
`setOptionValueWithSource`	`setOptionValueWithSource(key: undefined, value: undefined, source: undefined)`	`void`
`getOptionValueSource`	`getOptionValueSource(key: undefined)`	`void`
`getOptionValueSourceWithGlobals`	`getOptionValueSourceWithGlobals(key: undefined)`	`void`
`_prepareUserArgs`	`_prepareUserArgs(argv: undefined, parseOptions: undefined)`	`void`
`parse`	`parse(argv: undefined, parseOptions: undefined)`	`void`
`parseAsync`	`parseAsync(argv: undefined, parseOptions: undefined)`	`void`
`_prepareForParse`	`_prepareForParse()`	`void`

Method	Signature	Returns
`saveStateBeforeParse`	`saveStateBeforeParse()`	`void`
`restoreStateBeforeParse`	`restoreStateBeforeParse()`	`void`
`_checkForMissingExecutable`	`_checkForMissingExecutable(executableFile: undefined, executableDir: undefined, subcommandName: undefined)`	`void`
`_executeSubCommand`	`_executeSubCommand(subcommand: undefined, args: undefined)`	`void`
`_dispatchSubcommand`	`_dispatchSubcommand(commandName: undefined, operands: undefined, unknown: undefined)`	`void`

Where it refuses work

- `Command` stops the work with `Error` when `!cmd._name`.
- `Command` stops the work with `Error` when `previousArgument?.variadic`.
- `Command` stops the work with `Error` when `argument.required && argument.defaultValue !== undefined && argument.parseArg === undefin⋯`.
- `Command` stops the work with `Error` when `!allowedValues.includes(event)`.
- `Command` stops the work with `Error` when `typeof flags === 'object' && flags instanceof Option` — “To add an Option object use addOption() instead of option() or requiredOption()”.
- `Command` stops the work with `Error` when `this.parent && this._passThroughOptions && !this.parent._enablePositionalOptions`.

When something fails

- `Command` handles failure in 2 places: it logs it and continues in 1, and lets it reach the caller in 1.

Diagram

```
graph LR
  Command -->|command() / addCommand()| ChildCommand
  Command -->|createCommand()| NewCommand
  Command -->|createHelp()| Help
  Command -->|configureHelp()| HelpSettings
  Command -->|configureOutput()| OutputSettings
  Command -->|copyInheritedSettings()| InheritedSettings
  Command -->|showHelpAfterError()| ErrorHelp
```

```
  Command -->|showSuggestionAfterError()| ErrorSuggestion
  ChildCommand -->|_getCommandAndAncestors()| CommandHierarchy
```

Usage

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

const program = new Command();

program
  .name('tool')
  .description('Example command-line tool')
  .configureHelp({
    sortSubcommands: true,
  })
  .configureOutput({
    writeErr: (message) => process.stderr.write(message),
  })
  .showHelpAfterError()
  .showSuggestionAfterError();

program
  .command('serve')
  .description('Start the service')
  .action(() => {
    console.log('Service started');
  });

program.parse();
```

AI Coding Instructions

- Add subcommands through `command()` or `addCommand()` so they are attached to the command hierarchy.
- Override `createCommand()` when a subclass needs to create a custom command type for child commands.
- Configure help and output behavior before adding child commands when child commands should inherit those settings.
- Treat `_getCommandAndAncestors()` as an internal helper for hierarchy traversal; avoid calling it from application code.
- Keep `showHelpAfterError()` and `showSuggestionAfterError()` aligned with configured output handlers so error messages reach the expected stream.

Relationships

- IMPORTS → `Argument`
- IMPORTS → `humanReadableArgName`
- IMPORTS → `CommanderError`
- IMPORTS → `Help`
- IMPORTS → `Option`
- IMPORTS → `DualOptions`
- IMPORTS → `suggestSimilar`

Used by

1 reference from 1 file. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (1)

- `program` — `index.js`:7