CommanderError

August 25, 2026

CommanderError

Kind: Class

Source: `lib/error.js`

CommanderError class

`CommanderError` represents a command-processing failure with an associated exit code, error code, and message. It is thrown by command handling code so callers can report the failure and set the process exit status consistently.

Extends: `Error`

Diagram

```
graph LR
  Command[Command processing] --> Validation[Validation or action failure]
  Validation --> Error[CommanderError]
  Error --> Code[code]
  Error --> Message[message]
  Error --> ExitCode[exitCode]
  Error --> Handler[Error handler]
  Handler --> ProcessExit[Process exit status]
```

Usage

```
import { CommanderError } from './lib/error.js';

function abortCommand(exitCode: number, message: string): never {
  throw new CommanderError(
    exitCode,
    'commander.invalidArgument',
    message,
  );
}
```

```
try {
  abortCommand(process.exitCode, 'A required argument is missing.');
} catch (error) {
  if (error instanceof CommanderError) {
    console.error(error.message);
    process.exitCode = error.exitCode;
  } else {
    throw error;
  }
}
```

AI Coding Instructions

- Create `CommanderError` instances when command execution needs a known exit code and machine-readable error code.
- Preserve `code`, `exitCode`, and `message` when rethrowing or handling command errors.
- Check for `error instanceof CommanderError` before reading Commander-specific properties.
- Set `process.exitCode` from `error.exitCode` in top-level error handling rather than discarding the error status.
- Keep any original failure attached through `nestedError` when wrapping another error.

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