ReplFunction

August 17, 2026

ReplFunction

Kind: Class

Source: `packages/core/repl/repl-function.ts`

Part of: Core

`ReplFunction` represents an executable function exposed through the core REPL system. It provides the action implementation through `action()` and generates user-facing command documentation through `makeHelpMessage()`.

Methods

Method	Signature	Returns
`action`	`action(args: ActionParams)`	`ActionReturn`
`makeHelpMessage`	`makeHelpMessage()`	`string`

Properties

Property	Type
`fnDefinition`	`ReplFnDefinition`
`logger`	`Logger`

Diagram

```
graph LR
  User[REPL User] --> Parser[REPL Command Parser]
  Parser --> Function[ReplFunction]
  Function --> Help[makeHelpMessage()]
  Function --> Action[action()]
```

```
  Action --> Result[ActionReturn]
  Result --> User
```

Usage

```
import { ReplFunction } from "@core/repl/repl-function";

function runReplFunction(replFunction: ReplFunction) {
  // Display help when the function is requested without valid input.
  console.log(replFunction.makeHelpMessage());

  // Execute the function through the REPL runtime.
  const result = replFunction.action();

  return result;
}
```

AI Coding Instructions

- Keep command execution logic inside `action()` and return the expected `ActionReturn` value for the REPL runtime.
- Use `makeHelpMessage()` to provide concise, user-facing guidance for the function's syntax and behavior.
- Avoid writing directly to the console from `action()` unless the surrounding REPL conventions require it; prefer returning structured output.
- Ensure new REPL functions are registered through the relevant REPL command/function integration point.
- Keep help text aligned with the arguments and behavior implemented by `action()`.

How it works

`ReplFunction`

`ReplFunction` is an abstract base class for a REPL-native function. It is generic over the argument tuple accepted by `action` and its return type; both default to `Array<unknown>` and `any`, respectively. [repl-function.ts:6-9]

A subclass must define:

- `fnDefinition`, containing a function `name`, `description`, and `signature`; it may also declare aliases. The interface documents `name` as a valid JavaScript function

name and describes `signature` as TypeScript function-type-expression syntax. [repl-function.ts:10-11] [repl.interfaces.ts:4-19]

- `action(...args)`, the abstract method called for function invocation from the REPL. [repl-function.ts:19-22]

The constructor receives a `ReplContext`, stores it as a protected read-only `ctx`, and assigns `ctx.logger` to a protected read-only `logger`. [repl-function.ts:13-17] Built-in subclasses access `ctx` for application operations and output; for example, `GetReplFn.action` calls `ctx.app.get`, while `DebugReplFn` writes through `ctx.writeToStdout` and reports a missing module through `logger.error`. [native-functions/get-repl-fn.ts:14-16] [native-functions/debug-repl-fn.ts:15-36]

Relationships

- IMPORTS → `Logger`