

[Products](#) [Pricing](#) [Documentation](#)**npm**[Sign Up](#)[Sign In](#)[Search](#)**inquirer** 8.2.2 • [Public](#) • Published 9 days ago [Readme](#) [Explore](#) [BETA](#) [14 Dependencies](#) [35.043 Dependents](#) [115 Versions](#)

Inquirer.js

[npm package](#) [8.2.2](#) [codecov](#) [88%](#) [license scan](#) [passing](#)

A collection of common interactive command line user interfaces.

Table of Contents

1. **Documentation**
 1. **Installation**
 2. **Examples**
 3. **Methods**
 4. **Objects**
 5. **Questions**
 6. **Answers**
 7. **Separator**

8. **Prompt Types**
2. **User Interfaces and Layouts**
 1. **Reactive Interface**
3. **Support**
4. **Known issues**
5. **News**
6. **Contributing**
7. **License**
8. **Plugins**

Goal and Philosophy

`Inquirer.js` strives to be an easily embeddable and beautiful command line interface for **Node.js** (and perhaps the "CLI **Xanadu**").

`Inquirer.js` should ease the process of

- providing *error feedback*
- *asking questions*
- *parsing* input
- *validating* answers
- managing *hierarchical prompts*

Note: `Inquirer.js` provides the user interface and the inquiry session flow. If you're searching for a full blown command line program utility, then check out **commander**, **vorpal** or **args**.

Documentation

Installation

```
npm install inquirer
```

```
var inquirer = require('inquirer');  
inquirer
```

```
.prompt([
  /* Pass your questions in here */
])
.then((answers) => {
  // Use user feedback for... whatever!!
})
.catch((error) => {
  if (error.isTtyError) {
    // Prompt couldn't be rendered in the current environr
  } else {
    // Something else went wrong
  }
});
```

Examples (Run it and see it)

Check out the `packages/inquirer/examples/` folder for code and interface examples.

```
node packages/inquirer/examples/pizza.js
node packages/inquirer/examples/checkbox.js
# etc...
```

Methods

`inquirer.prompt(questions, answers) -> promise`

Launch the prompt interface (inquiry session)

- **questions** (Array) containing **Question Object** (using the **reactive interface**, you can also pass a `Rx.Observable` instance)
- **answers** (object) contains values of already answered questions. Inquirer will avoid asking answers already provided here. Defaults `{}`.
- returns a **Promise**

`inquirer.registerPrompt(name, prompt)`

Register prompt plugins under `name`.

- **name** (string) name of the this new prompt. (used for question type)
- **prompt** (object) the prompt object itself (the plugin)

`inquirer.createPromptModule()` -> prompt function

Create a self contained inquirer module. If you don't want to affect other libraries that also rely on inquirer when you overwrite or add new prompt types.

```
var prompt = inquirer.createPromptModule();
```

```
prompt(questions).then(/* ... */);
```

Objects

Question

A question object is a hash containing question related values:

- **type**: (String) Type of the prompt. Defaults: `input` - Possible values: `input`, `number`, `confirm`, `list`, `rawlist`, `expand`, `checkbox`, `password`, `editor`
- **name**: (String) The name to use when storing the answer in the answers hash. If the name contains periods, it will define a path in the answers hash.
- **message**: (String|Function) The question to print. If defined as a function, the first parameter will be the current inquirer session answers. Defaults to the value of `name` (followed by a colon).
- **default**: (String|Number|Boolean|Array|Function) Default value(s) to use if nothing is entered, or a function that returns the default value(s). If defined as a function, the first parameter will be the current inquirer session answers.
- **choices**: (Array|Function) Choices array or a function returning a choices array. If defined as a function, the first parameter will be the current inquirer session answers. Array values can be simple `numbers`, `strings`, or `objects` containing a `name` (to display in list), a `value` (to save in the answers hash), and a `short` (to display after selection) properties. The choices array can also contain a **Separator**.
- **validate**: (Function) Receive the user input and answers hash. Should return `true` if the value is valid, and an error message (`String`) otherwise. If `false` is returned, a default error message is provided.
- **filter**: (Function) Receive the user input and answers hash. Returns the filtered value to be used inside the program. The value returned will be added to the *Answers* hash.

- **transformer:** (Function) Receive the user input, answers hash and option flags, and return a transformed value to display to the user. The transformation only impacts what is shown while editing. It does not modify the answers hash.
- **when:** (Function, Boolean) Receive the current user answers hash and should return `true` or `false` depending on whether or not this question should be asked. The value can also be a simple boolean.
- **pageSize:** (Number) Change the number of lines that will be rendered when using `list`, `rawList`, `expand` or `checkbox`.
- **prefix:** (String) Change the default *prefix* message.
- **suffix:** (String) Change the default *suffix* message.
- **askAnswered:** (Boolean) Force to prompt the question if the answer already exists.
- **loop:** (Boolean) Enable list looping. Defaults: `true`

`default`, `choices` (if defined as functions), `validate`, `filter` and `when` functions can be called asynchronously. Either return a promise or use `this.async()` to get a callback you'll call with the final value.

```
{
  /* Preferred way: with promise */
  filter() {
    return new Promise(/* etc... */);
  },

  /* Legacy way: with this.async */
  validate: function (input) {
    // Declare function as asynchronous, and save the done c
    var done = this.async();

    // Do async stuff
    setTimeout(function() {
      if (typeof input !== 'number') {
        // Pass the return value in the done callback
        done('You need to provide a number');
        return;
      }
      // Pass the return value in the done callback
```

```

        done(null, true);
    }, 3000);
}
}

```

Answers

A key/value hash containing the client answers in each prompt.

- **Key** The `name` property of the *question* object
- **Value** (Depends on the prompt)
 - `confirm`:(Boolean)
 - `input` :User input (filtered if `filter` is defined) (String)
 - `number` :User input (filtered if `filter` is defined) (Number)
 - `rawlist` , `list` :Selected choice value (or name if no value specified) (String)

Separator

A separator can be added to any `choices` array:

```

// In the question object
choices: [ "Choice A", new inquirer.Separator(), "choice B" ]

```

```

// Which'll be displayed this way
[?] What do you want to do?
> Order a pizza
  Make a reservation
  -----
  Ask opening hours
  Talk to the receptionist

```

The constructor takes a facultative `String` value that'll be use as the separator. If omitted, the separator will be `-----` .

Separator instances have a property `type` equal to `separator` . This should allow tools façading Inquirer interface from detecting separator types in lists.

Prompt types

Note:: *allowed options written inside square brackets ([]) are optional. Others are required.*

List- {type: 'list'}

Take type, name, message, choices[, default, filter, loop] properties.
(Note: default must be set to the index or value of one of the entries in choices)

```
~/Documents/oss/Inquirer.js/examples master*  
> node list.js  
? What do you want to do? Order a pizza  
? What size do you need?  
  Jumbo  
  Large  
> Standard  
  Medium  
  Small  
  Micro
```

Raw List- {type: 'rawlist'}

Take type, name, message, choices[, default, filter, loop] properties.
(Note: default must be set to the index of one of the entries in choices)

```
~/Documents/oss/Inquirer.js master*  
> node examples/rawlist.js  
? What do you want to do? Order a pizza  
? What size do you need  
  1) Jumbo  
  2) Large  
  3) Standard  
  4) Medium  
  5) Small  
  6) Micro _
```

Expand- {type: 'expand'}

Take type, name, message, choices[, default] properties. Note: default must be the index of the desired default selection of the array. If default key not provided, then help will be used as default choice

Note that the choices object will take an extra parameter called key for the expand prompt. This parameter must be a single (lowercased) character. The h

option is added by the prompt and shouldn't be defined by the user.

See `examples/expand.js` for a running example.

```
~/Documents/oss/Inquirer.js master*  
> node examples/expand.js  
? Conflict on `file.js`: (yadxH) y  
>> Overwrite
```

```
~/Documents/oss/Inquirer.js master*  
> node examples/expand.js  
? Conflict on `file.js`: (yadxH)  
y) Overwrite  
a) Overwrite this one and all next  
d) Show diff  
-----  
x) Abort  
h) Help, list all options  
Answer: d
```

Checkbox - {type: 'checkbox'}

Take `type`, `name`, `message`, `choices` [, `filter`, `validate`, `default`, `loop`] properties. `default` is expected to be an Array of the checked choices value.

Choices marked as `{checked: true}` will be checked by default.

Choices whose property `disabled` is truthy will be unselectable. If `disabled` is a string, then the string will be outputted next to the disabled choice, otherwise it'll default to "Disabled". The `disabled` property can also be a synchronous function receiving the current answers as argument and returning a boolean or a string.

```
> node examples/checkbox.js  
? Select toppings  
  ● Cheddar  
  ○ Parmesan  
  = The usual =  
  >● Mushroom  
  ○ Tomato  
  = The extras =  
  ○ Pineapple  
(Move up and down to reveal more choices)
```


Confirm- {type: 'confirm'}

Take type , name , message ,[default] properties. default is expected to be a boolean if used.

```
~/Documents/oss/Inquirer.js master*  
> node examples/pizza.js  
Hi, welcome to Node Pizza  
? Is this for delivery? (y/N) █
```

Input- {type: 'input'}

Take type , name , message [, default , filter , validate , transformer] properties.

```
~/Documents/oss/Inquirer.js master*  
> node examples/input.js  
? What's your first name Simon  
? What's your last name Doe  
? What's your phone number █  
>> Please enter a valid phone number
```

Input- {type: 'number'}

Take type , name , message [, default , filter , validate , transformer] properties.

Password- {type: 'password'}

Take type , name , message , mask ,[, default , filter , validate] properties.

```
~/Documents/oss/Inquirer.js master*  
> node examples/password.js  
? Enter a password [hidden]  
? Enter a masked password *****
```

Note that `mask` is required to hide the actual user input.

Editor- `{type: 'editor'}`

Take `type`, `name`, `message` [, `default`, `filter`, `validate`, `postfix`] properties

Launches an instance of the users preferred editor on a temporary file. Once the user exits their editor, the contents of the temporary file are read in as the result. The editor to use is determined by reading the `$VISUAL` or `$EDITOR` environment variables. If neither of those are present, notepad (on Windows) or vim (Linux or Mac) is used.

The `postfix` property is useful if you want to provide an extension.

Use in Non-Interactive Environments

`prompt()` requires that it is run in an interactive environment. (I.e. **One where `process.stdin.isTTY` is true**). If `prompt()` is invoked outside of such an environment, then `prompt()` will return a rejected promise with an error. For convenience, the error will have a `isTtyError` property to programmatically indicate the cause.

User Interfaces and layouts

Along with the prompts, Inquirer offers some basic text UI.

Bottom Bar- `inquirer.ui.BottomBar`

This UI present a fixed text at the bottom of a free text zone. This is useful to keep a message to the bottom of the screen while outputting command outputs on the higher section.

```
var ui = new inquirer.ui.BottomBar();

// pipe a Stream to the log zone
outputStream.pipe(ui.log);

// Or simply write output
ui.log.write('something just happened. ');
ui.log.write('Almost over, standby!');

// During processing, update the bottom bar content to displ
// or output a progress bar, etc
ui.updateBottomBar('new bottom bar content');
```

Reactive interface

Internally, Inquirer uses the **JS reactive extension** to handle events and async flows.

This mean you can take advantage of this feature to provide more advanced flows. For example, you can dynamically add questions to be asked:

```
var prompts = new Rx.Subject();
inquirer.prompt(prompts);

// At some point in the future, push new questions
prompts.next({
  /* question... */
});
prompts.next({
  /* question... */
});

// When you're done
prompts.complete();
```

And using the return value `process` property, you can access more fine grained callbacks:

```
inquirer.prompt(prompts).ui.process.subscribe(onEachAnswer,
```

Support (OS Terminals)

You should expect mostly good support for the CLI below. This does not mean we won't look at issues found on other command line - feel free to report any!

- **Mac OS:**
 - Terminal.app
 - iTerm
- **Windows (Known issues):**
 - ConEmu
 - cmd.exe
 - Powershell
 - Cygwin
- **Linux (Ubuntu, openSUSE, Arch Linux, etc):**
 - gnome-terminal (Terminal GNOME)
 - konsole

Known issues

Running Inquirer together with network streams in Windows platform inside some terminals can result in process hang. Workaround: run inside another terminal. Please refer to the <https://github.com/nodejs/node/issues/21771>

Calling a node script that uses Inquirer from grunt-exec can cause the program to crash. To fix this, add to your grunt-exec config `stdio: 'inherit'`. Please refer to <https://github.com/jharding/grunt-exec/issues/85>

News on the march (Release notes)

Please refer to the [GitHub releases section for the changelog](#)

Contributing

Unit test Unit test are written in **Mocha**. Please add a unit test for every new feature or bug fix. `npm test` to run the test suite.

Documentation Add documentation for every API change. Feel free to send typo fixes and better docs!

We're looking to offer good support for multiple prompts and environments. If you want to help, we'd like to keep a list of testers for each terminal/OS so we can contact you and get feedback before release. Let us know if you want to be added to the list (just tweet to **@vaxilart**) or just add your name to **the wiki**

License

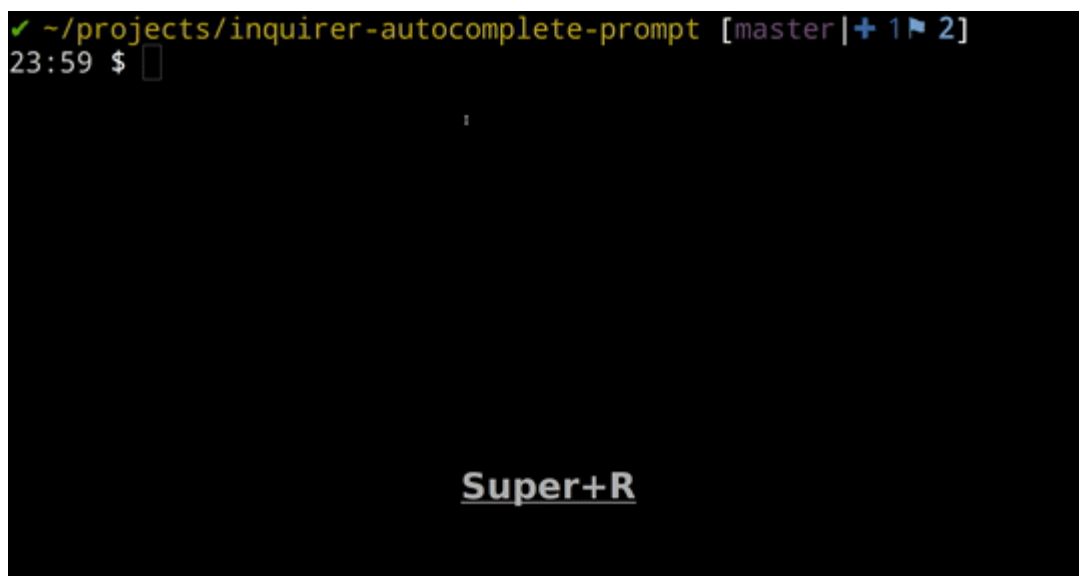
Copyright (c) 2016 Simon Boudrias (twitter: **@vaxilart**) Licensed under the MIT license.

Plugins

Prompts

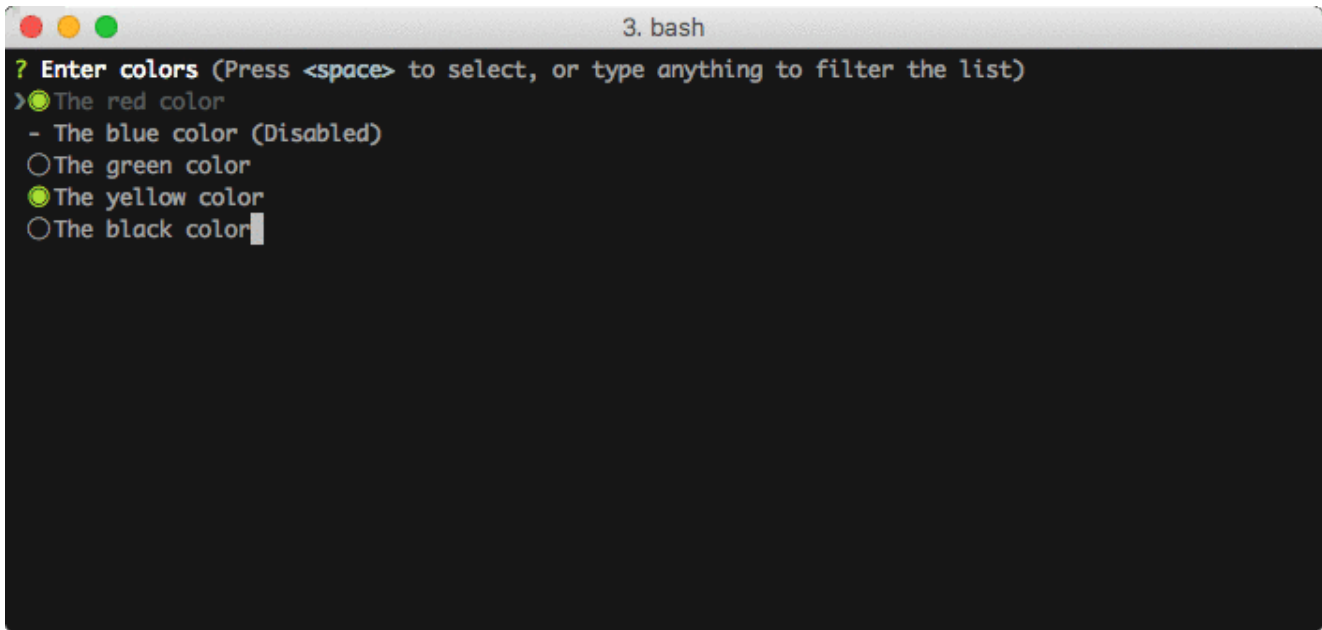
autocomplete

Presents a list of options as the user types, compatible with other packages such as fuzzy (for search)



checkbox-plus

Checkbox list with autocomplete and other additions



```
3. bash
? Enter colors (Press <space> to select, or type anything to filter the list)
> ☒ The red color
  - The blue color (Disabled)
  ☐ The green color
  ☒ The yellow color
  ☐ The black color
```

inquirer-date-prompt

Customizable date/time selector with localization support



```
examples $
```

datetime

Customizable date/time selector using both number pad and arrow keys



```
? When would you like a table? 1/2/01 5:00 PM
```

inquirer-select-line

Prompt for selecting index in array where add new element

```
? Where add line? (Use arrow keys)
> INSERT HERE
first
second
third
fourth
```

command

Simple prompt with command history and dynamic autocomplete

inquirer-fuzzy-path

Prompt for fuzzy file/directory selection.

inquirer-emoji

Prompt for inputting emojis.

```
? Input your favorite emoji: (Use arrow keys or type to search)
> 😊
  😞
  🙄
  🙊
  🙋
  😊
  😞
  🙊
(Move up and down to reveal more choices)
```

inquirer-chalk-pipe

Prompt for input chalk-pipe style strings

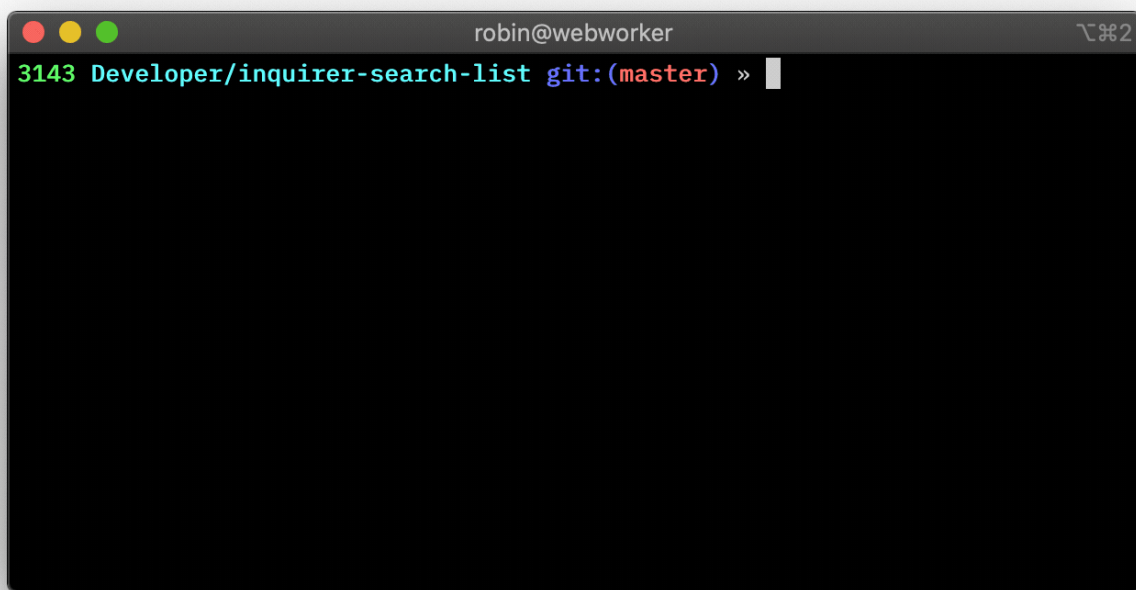
inquirer-chalk-pipe

inquirer-search-checkbox

Searchable Inquirer checkbox

inquirer-search-list

Searchable Inquirer list



inquirer-prompt-suggest

Inquirer prompt for your less creative users.

```
? Enter a name for your kitten: (Use tab for suggestions) |
```

inquirer-s3

An S3 object selector for Inquirer.


```
~/D/G/inquirer-s3 >>> node examples/s3-object-list.js
? Which S3 object would you like to select?
  Current directory: Buckets/
> inquirer-demo-bucket-1
  inquirer-demo-bucket-2
  inquirer-demo-bucket-3
  my-inquirer-assets
```

inquirer-autosubmit-prompt

Auto submit based on your current input, saving one extra enter

inquirer-file-tree-selection-prompt

Inquirer prompt for to select a file or directory in file tree

```
1. anchao01@126251i: ~/code/self-project/inquirer-file-tree-selection (zsh)
-----
~/code/self-project/inquirer-file-tree-selection(master*) » node ./example/demo.js|
```

inquirer-table-prompt

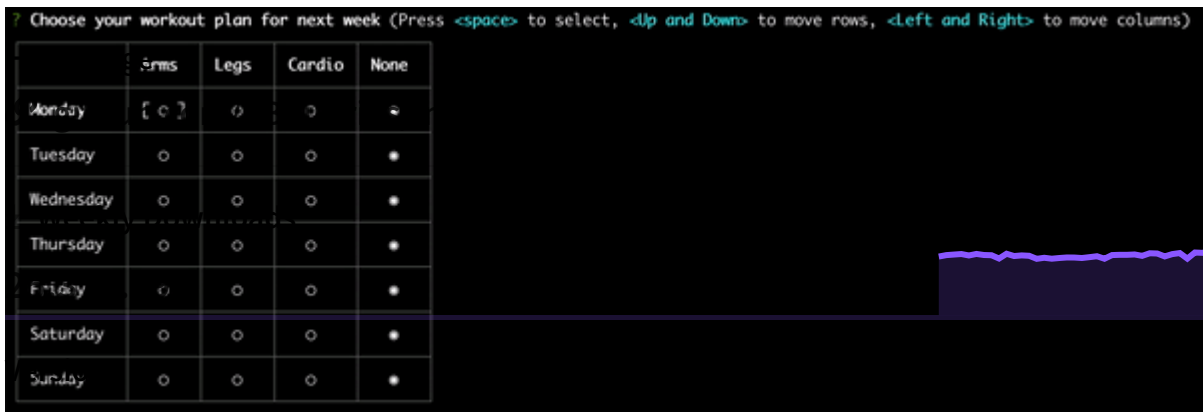
install

A table-like prompt for Inquirer.

```
> npm i inquirer
```

Repository

 github.com/SBoudrias/Inquirer.js



8.2.2

MIT

Unpacked Size

Keywords

86.6 kB

Total Files

26

command prompt stdin cli tty menu

Issues

Pull Requests

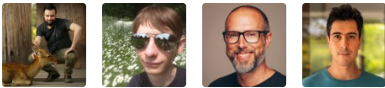
209

10

Last publish

9 days ago

Collaborators



>Try on RunKit

🚩Report malware



Support

[Help](#)

[Advisories](#)

[Status](#)

[Contact npm](#)

Company

[About](#)

[Blog](#)

[Press](#)

Terms & Policies

[Policies](#)

[Terms of Use](#)

[Code of Conduct](#)

[Privacy](#)