REPL Terminal in Node.js

REPL is an acronym for Read Eval Print Loop (understandably: Reading - Evaluating - Printing - Repeating) and it represents the computer environment like the console screen in the Linux shell where you can type the command line and the system The system will return the results. Node.js also has a REPL environment.

REPL is an acronym for Read Eval Print Loop (understandably: Reading - Evaluating - Printing - Repeating) and it represents the computer environment like the console screen in the Linux shell where you can type the command line and the system The system will return the results. Node.js also has a REPL environment. It to perform the desired tasks:

Read : Read user input information, convert into Javascript data and store in memory.

Eval : Evaluate these data structures.

Print : Print the results.

Loop : Repeat the command line until the user types **ctrl-c** twice.

The REPL feature of Node js is very useful when you use Node.js for the purpose of debugging code.

Start with REPL in Node.js

REPL can start by simply on the shell / console screen without using any of the following parameters:

```
$ node
```

You will see the REPL> prompt prompt. Here, you can type any command in Node.js.

```
$ node
>
```

Simple expressions in Node.js

Below I introduce some simple expressions that can be used at the REPL command prompt in Node.js:

```
$ node
> 1 + 3
4
> 1 + (2 * 3) - 4
3
>
```

Use variables in Node.js

You can use variables to store values ??and then print the value of the same variable as in traditional code snippets. If not using the **var** keyword, the value will be stored in the variable and printed. Meanwhile if the **var** keyword is used, the value is stored and may not be printed. You can print variables using console.log ().

```
$ node
> x = 10
ten
> var y = 10
undefined
> x + y
20
> console.log ("Hello World")
Hello World
undefined
```

Expressions on multiple lines in Node.js

The REPL node supports expressions that reside on many similar lines as in Javascript. Together check the do-while expression in the following action:

```
$ node
> var x = 0
undefined
> due to {
. x ++;
. console.log ("x:" + x);
.} while (x 5);
x: 1
x: 2
x: 3
x: 4
x: 5
undefined
>
```

The dashes . display automatically when you press the Enter key after the opening parenthesis. Node.js will automatically check to see if the expression is continuing.

The variable has an underscore in Node.js

You can use underscores _ to get the final result of the calculation:

```
$ node
> var x = 10
undefined
> var y = 20
undefined
```

```
> x + y
30
> var sum = _
undefined
> console.log (sum)
30
undefined
>
```

Introducing some REPL commands in Node.js

ctrl + c - End the current command.

Press ctrl + c twice - End Node REPL.

ctrl + d - End of the REPL node.

Up / Down arrow keys - View the history of commands, check the previous command and can modify previously edited commands.

Tab key - List of current commands.

.help - List all commands.

.break - Exit an expression that sits on multiple lines (eg, do-while).

.clear - Exit an expression that sits on multiple lines

.save ten_file - Save the current Node.js REPL session to a certain ten_file.

.load ten_file - Download the content of the current Node.js REPL ten_file.

Finish REPL in Node.js

As mentioned above, you need to press **ctr + c** twice to end the Node.js REPL.

```
$ node > (^C again to quit) >
```

According to Tutorialspoint

Last lesson: Hello World program in Node.js

Next article: NPM in Node.js

You finished reading the article "**REPL Terminal in Node.js**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.

© 2019 TipsMake.com