Node.js - REPL Terminal

REPL stands for Read Eval Print Loop and it represents a computer environment like a Windows console or Unix/Linux shell where a command is entered and the system responds with an output in an interactive mode. Node.js or **Node** comes bundled with a REPL environment. It performs the following tasks −

- **Read** − Reads user's input, parses the input into JavaScript data-structure, and stores in memory.
- **Eval** − Takes and evaluates the data structure.
- **Print** − Prints the result.
- **Loop** − Loops the above command until the user presses **ctrl-c** twice.

The REPL feature of Node is very useful in experimenting with Node.js codes and to debug JavaScript codes.

Online REPL Terminal

To simplify your learning, we have set up an easy to use Node.js REPL environment online, where you can practice Node.js syntax − Launch Node.js REPL Terminal

Starting REPL

REPL can be started by simply running **node** on shell/console without any arguments as follows.

```
$ node
```

You will see the REPL Command prompt > where you can type any Node.js command −

```
$ node
>
```

Simple Expression

Let's try a simple mathematics at the Node.js REPL command prompt −

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

Use Variables

You can make use variables to store values and print later like any conventional script. If **var** keyword is not used, then the value is stored in the variable and printed. Whereas if **var** keyword is used, then the value is stored but not printed. You can print variables using **console.log()**.

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

Multiline Expression

Node REPL supports multiline expression similar to JavaScript. Let's check the following do-while loop in action −

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

```
x: 3
x: 4
x: 5
undefined
>
```

... comes automatically when you press Enter after the opening bracket. Node automatically checks the continuity of expressions.

Underscore Variable

You can use underscore **(_)** to get the last result −

```
$ node
> var x = 10
undefined
> var y = 20
undefined
> x + y
30
> var sum = _
undefined
> console.log(sum)
30
undefined
>
```

REPL Commands

- **ctrl + c** − terminate the current command.
- **ctrl + c twice** − terminate the Node REPL.
- **ctrl + d** − terminate the Node REPL.
- **Up/Down Keys** − see command history and modify previous commands.
- **tab Keys** − list of current commands.
- **.help** − list of all commands.
- **.break** − exit from multiline expression.

- **.clear** − exit from multiline expression.
- **.save *filename*** − save the current Node REPL session to a file.
- **.load *filename*** − load file content in current Node REPL session.

Stopping REPL

As mentioned above, you will need to use **ctrl-c twice** to come out of Node.js REPL.

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