Node.JS

Practice – 1

Overview

*In this hands-on session you will learning to work with **Node.js**. Upon completion of this hands-on session, you should be able to:*

- *Version of **Node.js** and **npm***
- *Understand what is REPL*
    - *AutoComplete in REPL*
    - *REPL Special Commands*
    - *Special _ (Underscore) Variable*
- *Execute Node.js Programs*
- *Command Line Arguments*
- *Enhance Program output with*
    - *Format Specifier*
    - *Colouring*
- *Compute time spent of executing Node.js program*

Hands-on

1. Open a **terminal** and check the following:

   [a] Version of Node.js you are working on
   [b] Version of **Node Package Manager** (**npm**)

   Tip: If you are unsure how to open your terminal, google "`How to open terminal on your-operating-system`".

2. Start your **Node.js REPL** session and perform the following:

   [a] Try out some JavaScript code

   - Display **'Hello Node.js!'** in your console
   - Check the behaviour of **Arithmetic Operators**
   - Check the behaviour of **Relational/Comparison Operators**

   [b] Identify the **Special Commands** supported by **REPL** instance

   - Look into the .help command
   - Terminate your REPL session

[c] Once again start a new **REPL** session, using the **editor** feature of **REPL** define a function which takes **'name'** as its parameter and output's the following on its invocation/call

Assuming the name argument is **'Vijay'**, the output should look as shown below:

```
Hello Vijay
Welcome to <your_company_name>
Have a NICE day!!!
```

[d] Save the **REPL** session in a file by name **'first.js'**.

- Remember the folder/directory where you have saved the file.
- End the **REPL** session

[e] Once again start a new **REPL** session.

- Load the earlier saved file **'first.js'**
  Record your observation

[f] The shortcut key to clear the **REPL** session is **<CTRL>+L**

Try out some arithmetic expression and understand the behaviour of the **special _ (underscore) variable**

[g] Clear the **REPL** session and understand the following:

- Autocompletion using **TAB** key
- Type the name of the JavaScript class, like **'Number'** followed by a DOT and press the TAB key
- Thus, explore the JavaScript objects which prints all the properties and methods
- Similarly explore the global objects. Type **'global**.' and press **TAB** key
- End the **REPL** session

3. We can **import REPL** in a **JavaScript** file. To understand this behaviour, type the following JavaScript code in a file by name **'repl.js'**

```
// Using repl in JavaScript file
const local = require('repl');

local.start('$ ');  // Starting a REPL session
```

Save the file. Now execute this Node.js file.

- Observe, the **PROMPT** which is being displayed.
- Try out some JavaScript code and end the **REPL** session.

4. Node.js provides us with a **'process'** core module. It has the **exit()** method which will enable us to exit a Node.js program.

   Modify the earlier created **'repl.js'** program which will end the REPL session with a message **`'Exiting REPL...'`**

   HINT: Use the **'process'** modules **on()** method

   Moreover, check the **exit code** that you can give inside your terminal.

5. Create a Node.js program which will help us to understand the behaviour of **command line arguments.**

   HINT: The process module has the **'argv'** property will hold all the command line invocation arguments.

6. Node.js provides a **console** module which provides useful methods to interact with command line

   [a] Look into the list of console **properties** and **methods**

   [b] Write a Node.js program for the following:

   - Usage of format specifiers
   - Clearing the console
   - Counting elements

   [c] Write yet another Node.js program to understand the following:

   - The **'trace()**' method
   - Calculate the time spent to execute a JavaScript function

   [d] We are aware that **`console.log()`** is great for displaying message on the Console.

   - What happens when you perform REDIRECTION?
     Record your observation.
   - What should be done if we don't want STDERRs to be redirected.

   Write a Node.js program to understand the above topics

[e] An **Escape sequence** is a set of characters that identifies a colour.

Refer to the following URL:
`https://gist.github.com/iamnewton/8754917`
to get a list of some common colours.

HINT: It is generally a practice to display application messages in GREEN colour and error messages in RED colour

Write a Node.js program which will display text in different colours.