The Server Side Javascript

Prof. Rachana V. Modi

Node.js Console - REPL

- Node.js comes with virtual environment called REPL (Node shell).
- REPL → Read-Eval-Print-Loop
- It is a quick and easy way to test simple Node.js/JavaScript code.
- Javascript expressions

Node.js Console – REPL commands

REPL Command	Description
.help	Display help on all the commands
tab Keys	Display the list of all commands.
Up/Down Keys	See previous commands applied in REPL.
.editor	Enter editor mode in REPL.
.save filename	Save current Node REPL session to a file.
.load filename	Load the specified file in the current Node REPL session.
ctrl + c	Terminate the current command.
ctrl + c (twice)	Exit from the REPL.
ctrl + d	Exit from the REPL.
.break	Exit from multiline expression.
.clear	Exit from multiline expression.

Node.js Basics

- Primitive data types
- Arrays
- Buffer—Node.js super data type
- Object literals
- Functions
- Prototypal nature
- Conventions

Primitive Types

- String
- Number
- Boolean
- Undefined
- Null
- Symbol

Arrays

- Syntax:

```
var arr1 = new Array();
var arr2 = [];
```

- Methods & Properties:
  - push()
  - unshift()
  - indexing
  - length
  - splice()

Buffer

- Buffer class is designed to handle raw binary data.
- Fixed size
- Usage of buffer:
  - Reading data from file systems
  - Receiving packets over the network
- Syntax:

  Buffer.alloc(size, fill, encoding)
- From(): create a new buffer containing string
- Syntax: Buffer.from(object, encoding)

Object

- **Object** is collection of properties and methods which defines characteristics of object.

- **Ways to create object:**
  - Create objects using the Object function
  - Create objects using literal notation
  - Create objects using the object constructor
- **Example:**

```
var obj =
 {
   developerName: 'Ryan Dahl',
   language: 'Node.js'
  }
```

Object Function (New keyword)

Create objects using the Object function:

- **Syntax:**

```
objectName = new Object()
objectName.propertyName
```

- **Example:**

```
var myCar = new Object();
myCar.brand = 'Hyundai';
myCar.model = 'I10';
myCar.color = ‘white’;
```

Object Literal

Create objects using literal notation:

- A colon separates property name from value.
- A comma separates each name-value pair from the next.
- There should be no comma after the last name-value pair.

- **Example:**

```
var myCar =
{
   brand: 'Hyundai',
   model: 'i10',
   color: white
 };
```

Object Constructor

Create objects using Object Constructor:

- this keyword refers to the current object

Example:

```
function emp(id,name,salary)
{
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Mr, X",30000);
```

Functions

- A function is a block of organized and reusable code designed to perform a particular task.

- **Syntax:**

```
function functionName()
{
    // function body
    // optional return;
}
```

- **Example:**

```
function product(a, b)
{
    return a*b;
}
product(8,2);
```

Anonymous function

- Dynamically declared
- Without name

- Declaration of anonymous function:
  - Assignment to variable
  - Anonymous functions used as a parameter of another function
  - Immediately Invoked Function Expression (IIFE)

Anonymous function

Assignment to variable:

- Functions are always invoked using variable name.
- The function ends with a semicolon.

- **Example:**

```
var show = function () {
    console.log('Anonymous function');
};
Show();
```

Anonymous function

Callback function:

- Anonymous functions used as a parameter of another function.

- **Example:**

```
 setTimeout(function () {
    console.log('Execute later after 1 second') },
1000);
```

Anonymous function

Immediately invoked function execution:

- To execute function immediately after declaration.

- **Example:**

```
(function() {
    console.log('IIFE');
})();
```

Arrow function

- A shorthand for declaring anonymous functions.

- **Example:**

  Anonymous function:

  ```
  var add = function (a, b) {
      return a + b;
  };
  ```

  Anonymous function with Arrow:

  ```
  var add = (a, b)  => a + b;
  ```

Global Objects

- Global objects represents global scope.
- Node.js has a number of built-in global identifiers such as modules, functions, strings and object.
- A list of Node.js global objects:
  - __dirname
  - __filename
  - Console
  - Process
  - Buffer
  - setImmediate(callback[, arg][, ...])
  - setInterval(callback, delay[, arg][, ...])
  - setTimeout(callback, delay[, arg][, ...])
  - clearImmediate(immediateObject)
  - clearInterval(intervalObject)
  - clearTimeout(timeoutObject)

Process Object

- Each Node.js script runs in a process.
- It provides information about current process such as process id, architecture, platform, version, release, uptime and memory usage.

Process Properties

Property	Description
arch	returns process architecture: 'arm', 'ia32', or 'x64'
argv	returns commands line arguments as an array
env	returns user environment
pid	returns process id of the process
platform	returns platform of the process: 'darwin', 'freebsd', 'linux', 'sunos' or 'win32'
release	returns the metadata for the current node release
version	returns the node version
versions	returns the node version and its dependencies
stdin	readable stream for reading input from the user
stdout	writable stream, either synchronously or asynchronously.

Process Functions

Function	Description
cwd()	returns path of current working directory
hrtime()	returns the current high-resolution real time in a [seconds, nanoseconds] array
memoryUsage()	returns an object describing the memory usage of the Node process measured in bytes
process.kill(pid[, signal])	kill the process using pid.
uptime()	returns the Node.js process uptime in seconds.
exit()	terminate the process

Any query??