Node.js ZLIB

The Node.js zlib module is used to provide compression and decompression (zip and unzip) functionalities. It is implemented using Gzip and deflate/inflate.
The zlib module can be accessed using:

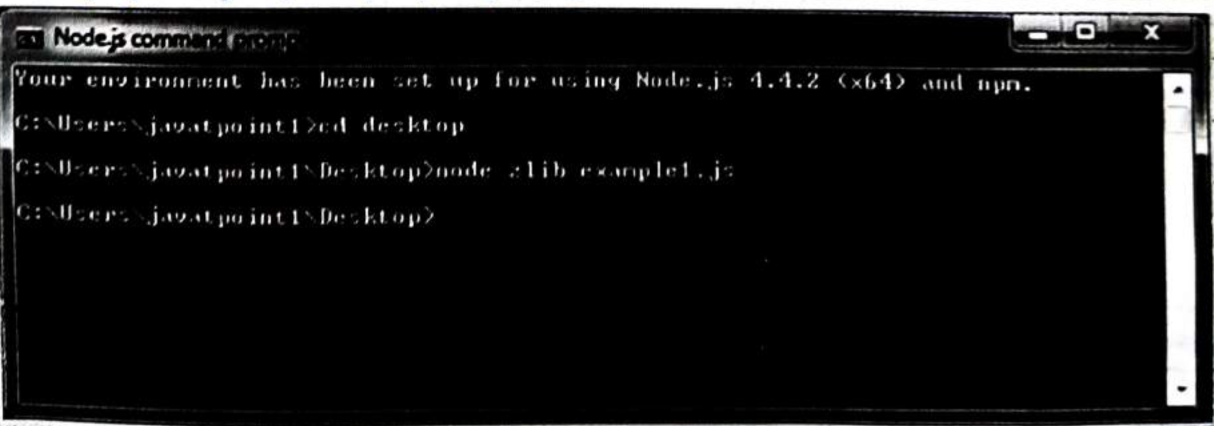
```
Constzlib = require('zlib');
```

Compressing and decompressing a file can be done by piping the Source stream data into a destination stream through zlib stream.

Node.js ZLIB Example : Compress File

File: zlib_example1.js

```
Const zlib    = require('zlib');
Constgzip     = zlib.CreateGzip();
Constfs       = require('fs');
Const inp     = fs.CreateReadStream('input txt');
Constout      = fs. CreateWritestream('input txt.gz');
inp. pipe(gzip). pipe(out)
```



```
Your environment has been set up for using Node.js 4.4.2 (x64) and npm.

C:\Users\javatpoint1>cd desktop

C:\Users\javatpoint1\Desktop>node zlib example1.js

C:\Users\javatpoint1\Desktop>
```

Node.js ZLIB Example: Decompress File

File: zlib_example2.js

```
Const zlib = require('zlib');
Const unzip = zlib.Create Unzip();
Const fs = require('fs');
Const inp = fs.createReadStream('input.txt.gz');
Const out = fs.createWritestream('input2.txt');
inp.pipe(unzip).pipe(out);
```

Node.js Assertion Testing

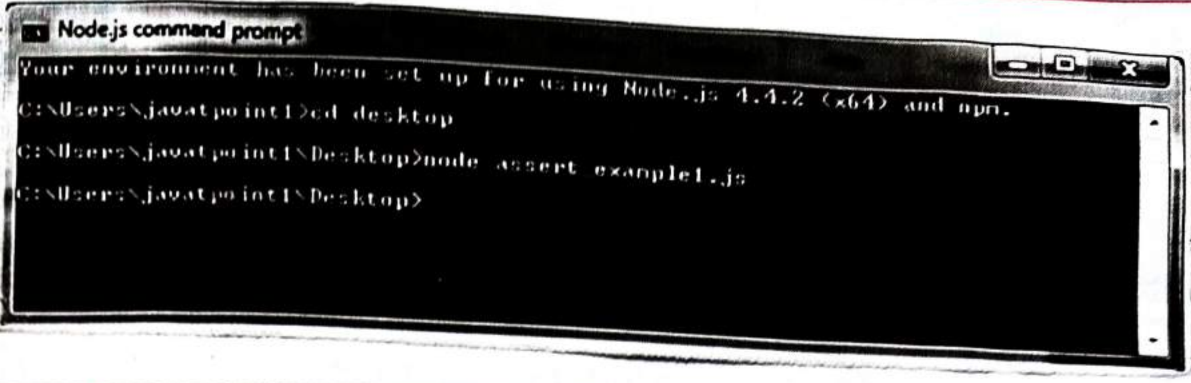
The Node.js Assert is the most elementary way to write tests. It provides no feedback when running your test unless one fails. The assert module provides a simple set of assertion tests that can be used to test invariants. The module is intended for internal use by Node.js, but can be used in application code via require('assert'). Howeverassert is not a testing framework and cannot be used as general purpose assertion library.

Node.js Assert Example

File: assert_example1.js

```
Var assert = require('assert');
function add (a,b){
   return a+b;
}
Var expected = add(1,2);
assert(expected===3,'one plus two is three');
```

It will not provide any output because the case is true. If you want to see output, you need to make the test fail.



```
Node.js command prompt
Your environment has been set up for using Node.js 4.4.2 (x64) and npm.
C:\Users\javatpoint1>cd desktop
C:\Users\javatpoint1\Desktop>node assert example1.js
C:\Users\javatpoint1\Desktop>
```

Node.js V8

What is V8

V8 is an open source JavaScript engine developed by the Chromium project for the Google Chrome Web browser. It is Written in C++. Now a days, it is used in many projects such as couchbase, mongoDB and Node.js.

V8 in Node.js

The Node.js V8 module represents interfaces and event specific to the Version of V8. It provides methods to get information about heap memory through V8.getHeapStatistics() and V8.getHeapSpacestatistics() methods. To use this module, you need to use require ('V8').

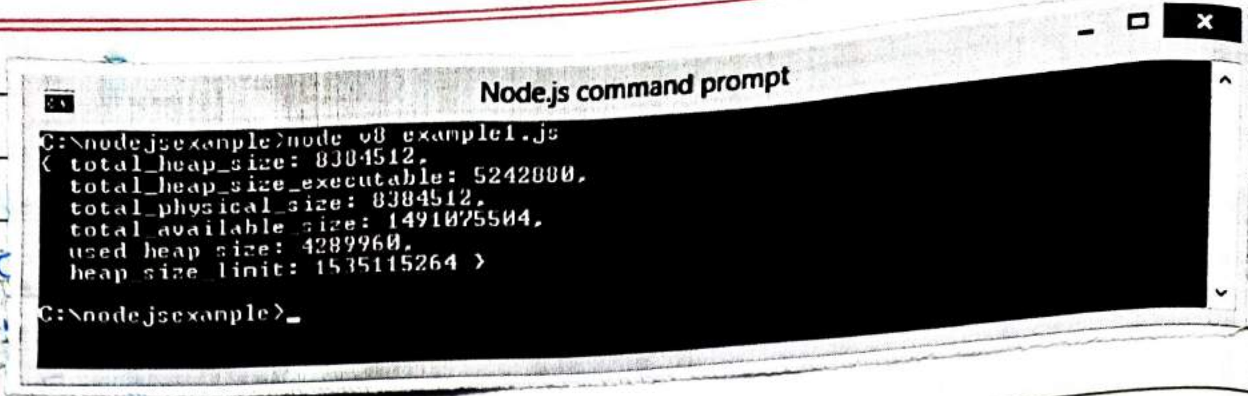
```
Const V8 = require ('V8');
```

Node.js V8.getHeapStatistics () Example

The V8.getHeapStatistics() method returns statistics about heap such as total heap size, used heap size, heap size limit, total available size etc.

File : V8_example1.js

```
Const V8 = require ('V8'):
console.log (V8.getHeapStatistics());
```



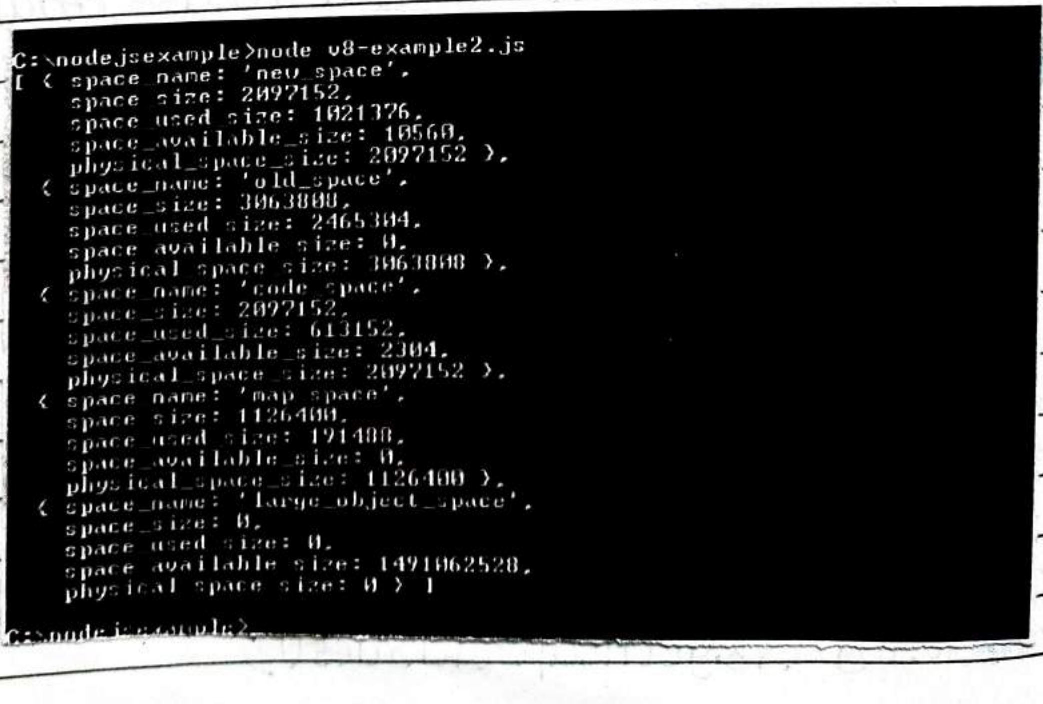
Node.js command prompt

```
C:\nodejsexample>node v8 example1.js
{ total_heap_size: 8384512,
  total_heap_size_executable: 5242880,
  total_physical_size: 8384512,
  total_available_size: 1491075504,
  used_heap_size: 4289960,
  heap_size_limit: 1535115264 }

C:\nodejsexample>_
```

Node.js V8. getHeapSpaceStatistics() Example

The V8 getHeapSpaceStatistics() returns statistics about heap space. It returns an array of 5 objects: new space, old space, code space, map space and large object space. Each object contains information about space name, space size, space used size, space available size and physical space size.

File: V8_example2.js
Const V8 = require('V8');
Console.log (V8.get HeapSpaceStatistics());



```
C:\nodejsexample>node v8-example2.js
[ { space_name: 'new_space',
    space_size: 2097152,
    space_used_size: 1021376,
    space_available_size: 10560,
    physical_space_size: 2097152 },
  { space_name: 'old_space',
    space_size: 3063808,
    space_used_size: 2465304,
    space_available_size: 0,
    physical_space_size: 3063808 },
  { space_name: 'code_space',
    space_size: 2097152,
    space_used_size: 613152,
    space_available_size: 2304,
    physical_space_size: 2097152 },
  { space_name: 'map_space',
    space_size: 1126400,
    space_used_size: 191488,
    space_available_size: 0,
    physical_space_size: 1126400 },
  { space_name: 'large_object_space',
    space_size: 0,
    space_used_size: 0,
    space_available_size: 1491062528,
    physical_space_size: 0 } ]

C:\nodejsexample>_
```

Memory limit of V8 in Node.js

Currently, by default V8 has a memory limit of 512mb on 32-bit and 1gb on 64-bit systems. You can raise the limit by setting -- max-old-space-sizo to a maximum of ~1gb for 32-bit and ~1.7gb for 64-bit systems. But it is recommended to split your single process into several Workers if you are bitting memory limits.

Node.js Callbacks

Callback is an asynchronous equivalent for a function. It is called at the completion of each task. In Node.js Callbacks are generally used. All APIs of Node are Written in a Way to supports Callbacks. For example: When a function start reading file, it returns the control to execution environment immediately So that the next instruction can be executed.

Blocking Code Example

Follow these steps:

1. Create a text file named input.txt having the following content:
   Javatpoint is an online platform providing self learning tutorials on different technologies, in a very simple language.

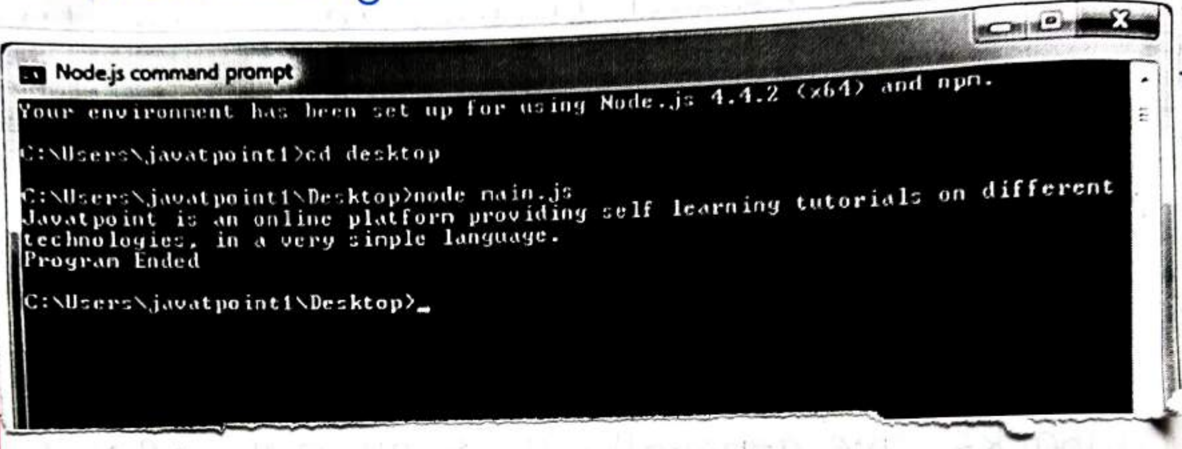
2. Create a JavaScript file named main.js having the following code:

```
Var fs = require("fs");
Var data = fs.readFileSync('input.txt');
```

```
Console.log (data.toString());
Console.log ("program Ended");
```

7 Open the Node.js Command prompt and execute the following code:

```
Ndde main.js
```



```
Node.js command prompt
Your environment has been set up for using Node.js 4.4.2 (x64) and npm.

C:\Users\javatpoint1>cd desktop

C:\Users\javatpoint1\Desktop>node main.js
Javatpoint is an online platform providing self learning tutorials on different
technologies, in a very simple language.
Program Ended

C:\Users\javatpoint1\Desktop>_
```

Non Blocking Code Example

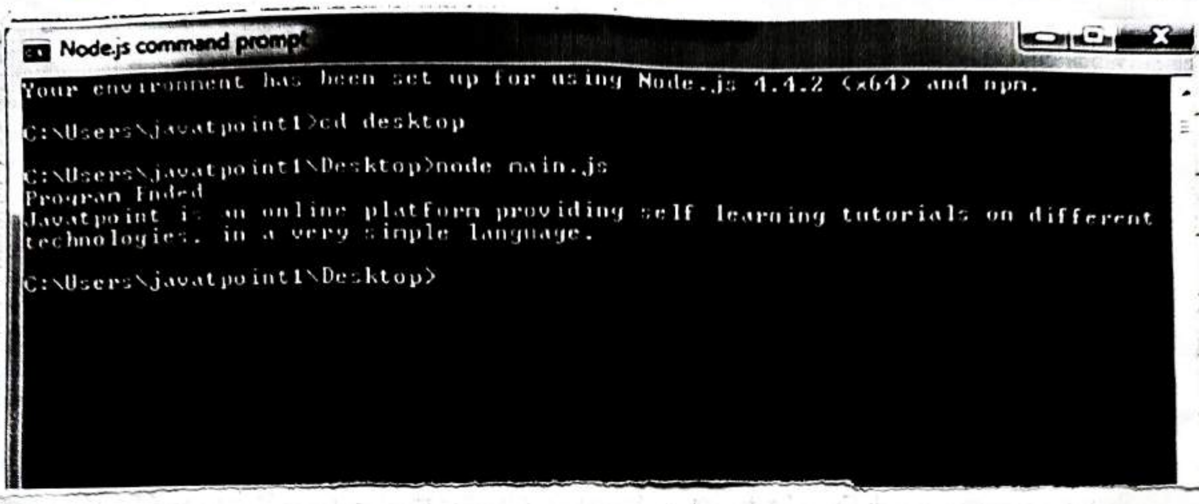
Follow these steps:

1. Create a text file named input.txt having the following content:
Javatpoint is an online platform providing self learning tutorials on different technologies, in a very simple language.

2. Create a JavaScript file named main.js having the following code:

```
Var fs = require ("fs");
fs.readFile('input.txt', function (err, data) {
    If(err) return console.error(err)
    Console.log (data.tostring());
});
Console.log ("Program Ended");
```

3. Open the Node.js Command prompt and execute the following Code.
node main.js



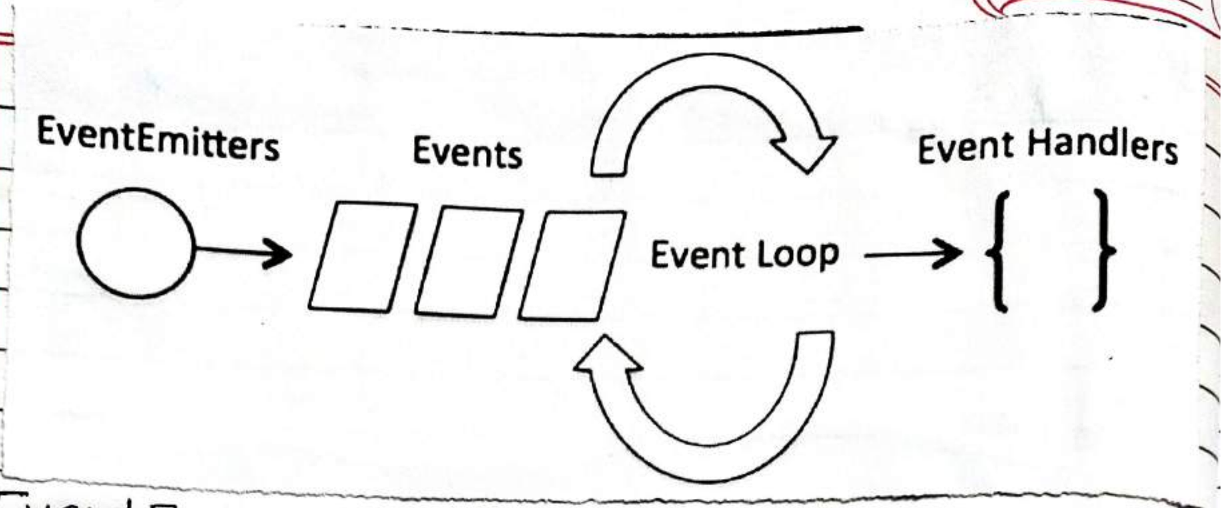
```
Your environment has been set up for using Node.js 4.4.2 (x64) and npm.

C:\Users\javatpoint1>cd desktop

C:\Users\javatpoint1\Desktop>node main.js
Program Ended
Javatpoint is an online platform providing self learning tutorials on different
technologies. in a very simple language.

C:\Users\javatpoint1\Desktop>
```

Node.js Events

In Node.js applications, Events and Callbacks concepts are used to provide Concurrency. As Node.js applications are single threaded and every API of Node js are asynchronous. so it uses async function to maintain the concurrency. Node uses observe pattern Node thread keeps an event loop and after the completion of any task, it fires the corresponding event Which signals the event listener function to get executed.

Event Driven Programming

Node.js uses event driven programming. It means as Soon as Node starts its server, it simply intiates its variables, declares function and then simply Waits for event to occur. It is the one of the reson Why Node.js is preety fast compared to other Similar technologies. There is a main loop in the event driven application that listens for events, and then triggers a callback function When one of those events is detected.



EventEmitter class to bind event and event listener:

```
// Import events module
Var events = require('events');
// Create an eventEmitter object
Var eventEmitter = new events. Event Emitter();
```

To bind event handler with an event:

```
// Bind event and even handler as follows
eventEmitter.on('event Name', eventHandler);
```

To fire an event:

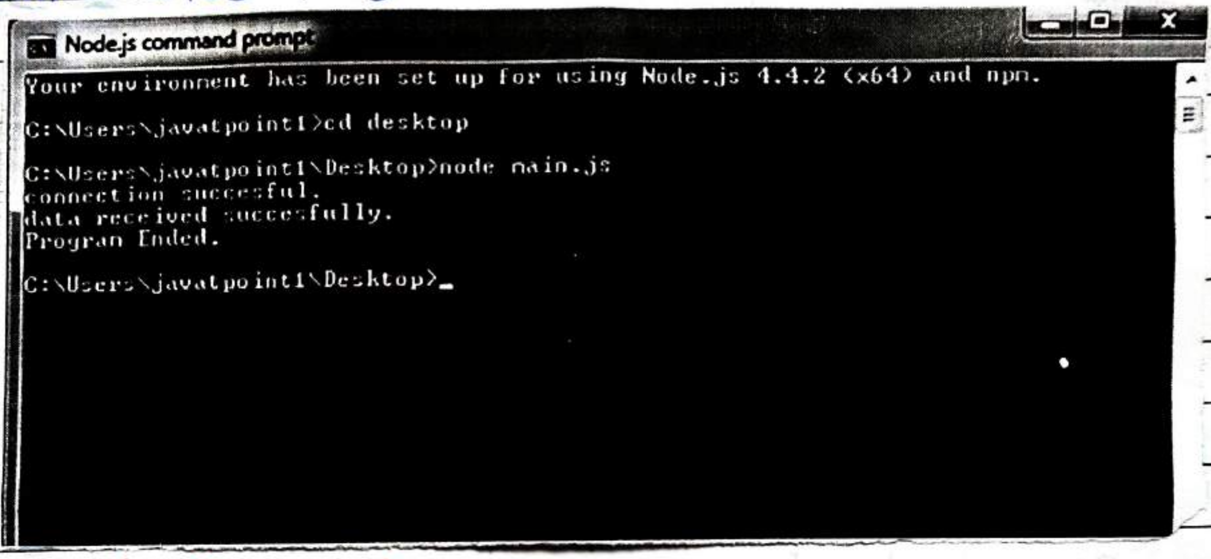
```
// Fire an event
eventEmitter.emit('eventName');
```

Node.js Event Example

File: main.js

```
// Import events module
Var events = require('events');
// Create an eventEmitter object
Var eventEmitter = new events.EventEmitter();
// Create an event handler as follows
Var ConnectHandler = function connected() {
   console.log('connection succesful.');
// Fire the data received event
eventEmitter.emit('data_received');
}
```

```
// Bind the Connection event With the handler,
eventEmitter.on('Connection', connectHandler);
// Bind the data_received event with the anonymous funct
eventEmitter.on('data received', function() {
   Console.log('data received succesfully.');
});
// Fire the connection event
eventEmitter.emit('connection');
Console.log("program Ended.");
```



```
Your environment has been set up for using Node.js 4.4.2 (x64) and npm.

C:\Users\javatpoint1>cd desktop

C:\Users\javatpoint1\Desktop>node main.js
connection succesful.
data received succesfully.
Program Ended.

C:\Users\javatpoint1\Desktop>_
```

Node.js Punycode

What is Punycode

Punycode is an encoding syntax which is used to convert Unicode (UTF-8) string of characters to basic ASCII string of characters. Since host names only understand ASCII characters so punycode is used. It is used as an Internationalizd domain name (IDN or IDNA)

Punycode in Node.js

Punycode.js is bundled with Node.js v0.6.2 and later version. If you want to use it with other Node.js versions then use npm to install punycode module first. You have to use require('punycode') to access it.

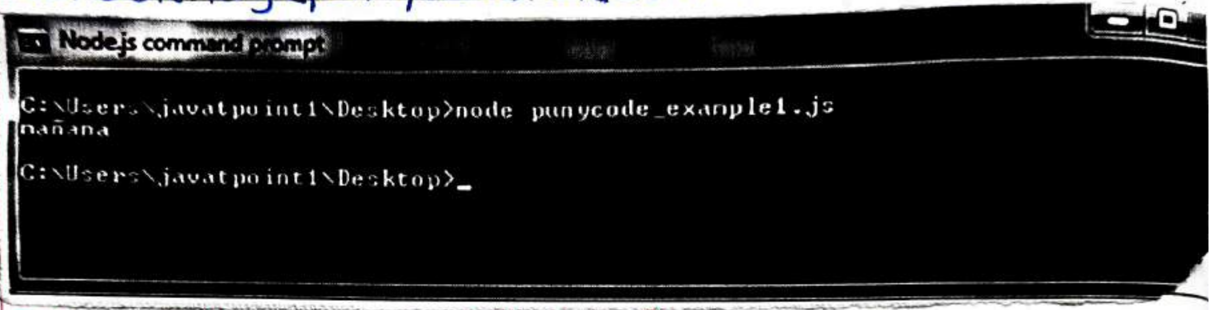
punycode = require('punycode');

punycode.decode(string)

It is used to convert a Punycode string of ASCII symbols to a string of Unicode symbols.

File: punycode_example1.js

```
punycode = require('punycode');
Console.log(punycode.decode('maana-pta'));
```



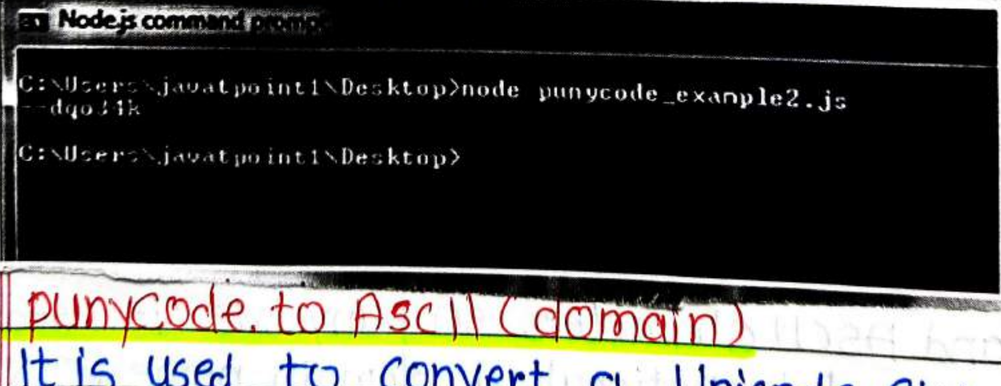
```
C:\Users\javatpoint1\Desktop>node punycode_example1.js
mañana

C:\Users\javatpoint1\Desktop>_
```

punycode.encode(string)

It is used to convert a string of Unicode symbols to a punycode string of ASCII symbols.

File: punycode_example2.js

```
punycode = require('punycode');
console. log ( punycode. encode('
```



```
C:\Users\javatpoint1\Desktop>node punycode_example2.js
--dqo34k

C:\Users\javatpoint1\Desktop>
```

punyCode.toASCII(domain)

It is used to convert a Unicode String representing a domain name to PunyCode. Only the non-ASCII part of the domain name is converted.

File: punycode_example3.js

```
punycode = require ('punyCode');
console. log (punyCode.toASCII('mañana.com'));
```