Concept of Buffer in Node.js

Net Javascript is Unicode encoded conveniently but not really good with binary data. When working with TCP streams or file systems, it is necessary to handle octal data streams. Node.js provides Buffer classes that allow raw data to be stored as an array of integers corresponding to external raw memory allocation V8 heap.

Net Javascript is Unicode encoded conveniently but not really good with binary data. When working with TCP streams or file systems, it is necessary to handle octal data streams. Node.js provides Buffer classes that allow raw data to be stored as an array of integers corresponding to external raw memory allocation V8 heap.

Buffer classes in Node.js are global classes and can be accessed in the application without declaring Buffer Modules by the require () method like other Modules.

Create buffers in Node.js

Buffers in Node.js can be built in many different ways.

Way 1

Syntax for creating a Buffer of size 10:

```
var  buf  = new Buffer ( 10 );
```

Method 2

Syntax to create a Buffer from a given array:

```
var  buf  = new Buffer ([ 10 , 20 , 30 , 40 , 50 ]);
```

Method 3

The syntax for creating a Buffer from a given string and with arbitrary encoding:

```
var  buf  = new Buffer ( "Hoc Nodejs tai QTM" , "utf-8" );
```

Although "utf8" is the default encoding, you can use other encodings such as "ascii", "utf8", "base64", .

Write data to Buffer in Node.js

Syntax

The syntax for writing a Buffer in Node.js is:

```
buf . write ( string [,  offset ][,  length ][,  encoding ])
```

Details about parameters

string - This is a string data written to the buffer.

offset - This is the index for the buffer to start recording there. The default value is 0.

length - The number of bytes recorded. The default is buffer.length.

encoding - Encoding is used. "utf8" is the default encoding.

Return value

The method will return the number of bytes written. If the memory in the buffer is not enough to satisfy the entire string, it will record part of the string.

The example illustrates how to write data to Buffer in Node.js

In this example, I use write () method to receive the parameter as string data to write that data to Buffer.

```
  buf = new Buffer ( 256 );  wool = buf .  write ( "Hoc Nodejs at QTM" );  conso
```

When the above program is executed will result:

```
Tong  so  byte  da ghi  : 22
```

Read data from Buffer in Node.js

Syntax

The syntax for reading data from Buffer in Node.js is as follows:

```
buf . toString ([ encoding ][,  start ][, end ])
```

Details about parameters

encoding - Is the encoding to use. Default encoding is 'utf8'.

start - Index to start reading operation, default value is 0.

end - Index to end the read operation, the default value is the length of the Buffer.

Return value

This method decodes and returns a string from the data encoded in the Buffer using a specific encoder.

The example illustrates how to read data from Buffer in Node.js

```
  buf = new Buffer ( 26 );  for ( var i = 0 ; i  26 ; i ++) { buf [ i ] = i + 97
```

When the above program is executed will result:

```
abcdefghijklmnopqrstuvwxyz abcde abcde abcde
```

Convert Buffer to JSON in Node.js

Syntax

To convert a Buffer in Node.js into a JSON object, you use the toJSON () method with the following syntax:

```
buf . toJSON ()
```

Return value

This method returns a JSON representation for the given Buffer object.

The example illustrates how to convert Buffer to JSON

```
var buf = new Buffer ( 'Simply Easy Learning' );  var json = buf .  toJSON ( bu
```

When the above program is executed will result:

```
[ 83 , 105 , 109 , 112 , 108 , 121 , 32 , 69 , 97 , 115 , 121 , 32 , 76 , 101 ,
```

Pair Buffers in Node.js

Syntax

To concatenate two or more Buffers into a Buffer in Node.js, you use concat () method as follows:

```
Buffer . concat ( list [,  totalLength ])
```

Details about parameters

list - Defines an array of Buffers used to pair into a Buffer.

totalLength - The total length of the buffers after being paired.

Return value

This method returns a new Buffer.

The example illustrates how to connect Buffers

```
var buffer1 = new Buffer ( 'The software provides a new version of the databas
```

When the above program is executed will result:

```
Noi  dung cua buffer3 la : QTM  cung cap loat bai huong dan mien phi cho tat ca
```

Compare Buffers in Node.js

Syntax

To compare two Buffers in Node.js, you use the compare () method as follows:

```
buf . compare ( otherBuffer );
```

Details about parameters

otherBuffer - Another Buffer to be compared to a Buffer named buf.

Return value

Returns a numerical value that represents this Buffer to stand before, after, or the same order as the other Buffer.

The example illustrates how to compare two Buffers in Node.js

```
  var buffer1 = new Buffer ( 'ABC' );  var buffer2 = new Buffer ( 'ABCD' );  var
```

When the above program is executed will result:

```
ABC dung truoc ABCD
```

Copy the Buffer in Node.js

Syntax

To copy the Buffer in Node.js, you use the copy () method as follows:

```
buf . copy ( targetBuffer [,  targetStart ][,  sourceStart ][,  sourceEnd ])
```

Details about parameters

targetBuffer - Buffer object, this is where the Buffer will be copied.

targetStart - Number format, default is 0.

sourceStart - Number format, the default is 0.

sourceEnd - Number format, the default is the length of the buffer.

Return value

This copy () method does not return any values.

The example illustrates how to copy Buffer

```
  var buffer1 = new Buffer ( 'ABC' );  // Why is the buffer buffer var buffer2 =
```

When the above program is executed will result:

```
Noi  dung cua buffer2 la :  ABC
```

Split the Buffer in Node.js

Syntax

To create a child Buffer of a Buffer in Node.js, use the slice () method as follows:

```
buf . slice ([ start ][, end ])
```

Details about parameters

start - Number format, the default value is 0.

end - Number format, the default value is buffer.length

Return value

Returns a new buffer that references the same memory area as the old Buffer.

For example

```
  var buffer1 = new Buffer ( 'VietNamVoDoi' );  // Divide the grapefruit buffer v
```

When the above program is executed will result:

```
Noi  dung cua buffer2 la : VietNam
```

Length Buffer in Node.js

Syntax

To get the length (equal to the byte value) of a Buffer in Node.js, you use the following length attribute:

```
buf . length ;
```

Return value

Returns the length in bytes of a Buffer.

For example

```
  var buffer = new Buffer ( 'VietNamVoDoi' );  // Due to the persistent buffer co
```

When the above program is executed will result:

```
Do  dai cua buffer la : 12
```

According to Tutorialspoint

Previous article: Event Emitter in Node.js

Next lesson: Stream in Node.js

You finished reading the article "**Concept of Buffer 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.
