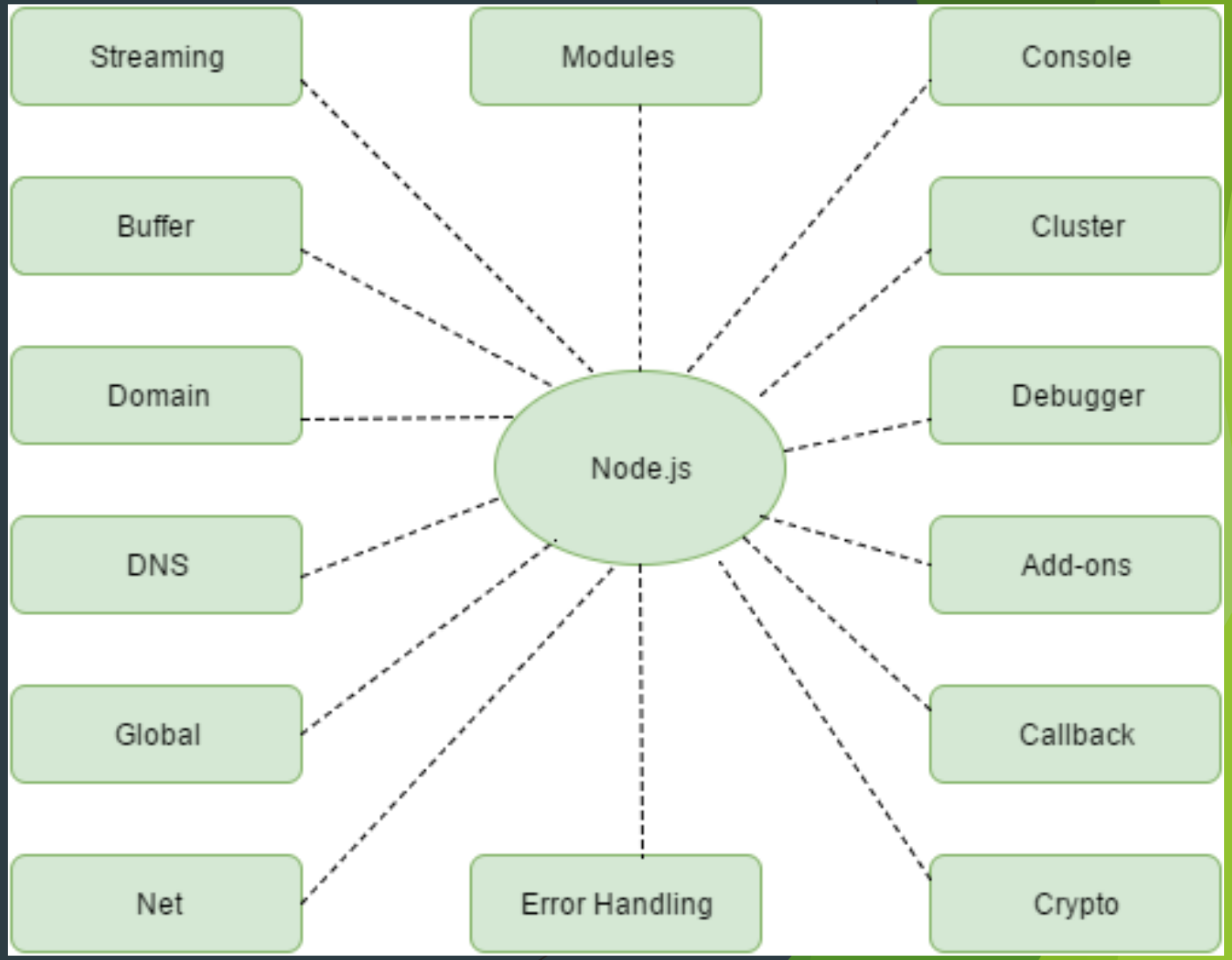
Node.js

Global - 김근형 강사

내장 객체와 내장 모듈

- 내장 객체와 내장 모듈



내장 객체와 내장 모듈

- 내장 객체와 내장 모듈
  - node.js 에서는 기본적으로 제공하는 내장 객체와 내장 모듈이 존재한다.
  - 해당 객체와 모듈을 통해 기본적인 기능의 구현이 가능하다.
  - 하지만 나중에 되어서 내장 객체와 내장 모듈 만으로 구현이 힘들 수 있으며 이후 나중에 소개 될 npm 을 이용해 다른 외부 모듈을 가져올 수 있다.

Global

- Global 객체
  - Global 객체는 Node.js에서 사용되는 전역 객체이다.
  - 웹에서 사용하는 window 전역 객체와 같은 포지션에 해당한다.
  - Global은 window에서 제공하는 내용과 유사한 기능도 존재하지만 다른 기능 또한 존재한다.
  - Global은 window처럼 생략이 가능하다.

Global

- Global 내장 함수 및 객체 종류

Class: Buffer	module	clearInterval(intervalObject)	TextDecoder
__dirname	process	clearTimeout(timeoutObject)	TextEncoder
__filename	queueMicrotask(callback)	setImmediate(callback[, ...args])	URL
console	require()	setInterval(callback, delay[, ...args])	URLSearchParams
exports	clearImmediate(immediateObject)	setTimeout(callback, delay[, ...args])	WebAssembly

console

- console
  - 콘솔에서 문자열을 출력할 때 주로 쓰이며 일반적으로 웹에서 사용하는 consol과 기능이 거의 동일하다
  - 다음과 같은 기능을 제공한다.

함수	기능
console.log([data][, ...args])	console.log()는 콘솔 화면에 문자열을 출력하는 함수로, 서버의 상태를 서버관리자에게 알려주기 위해 콘솔에 로그를 출력하는 기능으로 사용한다.
console.info([data], [...])	console.log()와 동일하다
console.error([data], [...])	입력 받은 메시지를 표준 오류로 출력할 때 사용하는 함수.
console.warn([data], [...])	console.error()와 동일 기능 단 error보다 덜 중요한 경고성의 문구를 출력할 경우 사용
console.time(label)	작업 기간을 계산하는 데 사용할 수 있는 타이머를 시작한다. 타이머는 console.timeEnd()를 호출할 때 동일한 라벨을 사용하여 타이머를 중지하고 경과시간을 밀리초 단위로 출력할 수 있다.
console.timeEnd(label)	console.time()로 호출된 타이머를 중지하고 결과를 출력한다.

console

- console

함수	기능
console.timelog(label)	console.time() 이후 경과한 시간을 출력한다. 타이머는 끝나지 않고 계속 진행이 된다.
console.trace(label)	로깅 시 해당 스택을 추적하고자 할 경우 사용하는 함수. 디버깅을 할 경우 유용하게 사용이 가능하다.
console.assert(expression, [message])	조건값인 true/false에 따라 출력 여부를 다르게 할 경우 사용한다. 주로 web worker에서 사용이 되는 경우가 많다.
console.count([label])	console.count() 메서드는 특정 label 호출의 횟수를 세어 출력한다. 만약 label을 쓰지 않을 경우 label에 관계없이 그냥 카운트만 한다.
console.countReset([label])	특정 내부의 카운터를 리셋한다.
console.table(tabularData[, properties])	출력할 객체를 테이블 형태로 출력할 경우 사용할 수 있다.

console

- console 예제 - 1

```
console.log("이것은 console.log로 출력한 문자열 입니다.",
"이런식의 출력도 가능합니다");
console.info("이것은 console.info","로 출력한 문자열 입니다.");
console.error("이것은 console.error로 출력한 메시지 입니다.");
console.warn("이것은 console.warm으로 출력한 메시지 입니다.");
```

```
이것은 console.log로 출력한 문자열 입니다. 이런식의 출력도 가능합니다
이것은 console.info 로 출력한 문자열 입니다.
이것은 console.error로 출력한 메시지 입니다.
이것은 console.warm으로 출력한 메시지 입니다.
```

```
var i;
console.time("test for loop");
for (i = 0; i < 1000000; i++) {}
console.timeLog("test for loop");
for (i = 0; i < 1000000; i++) {}
console.timeLog("test for loop");
for (i = 0; i < 1000000; i++) {}
console.timeEnd("test for loop");

i = 0;
console.time("test while loop");
while (i < 1000000) {
  i++
}
console.timeEnd("test while loop");
```

```
test for loop: 9.777099609375ms
test for loop: 10.576ms
test for loop: 23.525146484375ms
test for loop: 23.621ms
test for loop: 27.232177734375ms
test for loop: 27.393ms
test while loop: 2.18115234375ms
test while loop: 2.344ms
```

console

- console 예제 - 2

```
function foo() {
    function bar() {
        console.trace("my word");
    }
    bar();
}

foo();
```

```
Trace: my word
    at bar (c:\Users\User\OneDrive\수업\수업\node.js\v2.0\source\ch03\03_console3.js:3:1
5)
    at foo (c:\Users\User\OneDrive\수업\수업\node.js\v2.0\source\ch03\03_console3.js:5:5)
    at Object.<anonymous> (c:\Users\User\OneDrive\수업\수업\node.js\v2.0\source\ch03\03_c
onsole3.js:8:3)
    at Module._compile (internal/modules/cjs/loader.js:1137:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1157:10)
    at Module.load (internal/modules/cjs/loader.js:985:32)
    at Function.Module._load (internal/modules/cjs/loader.js:878:14)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:71:12)
    at internal/main/run_main_module.js:17:47
Waiting for the debugger to disconnect...
Process exited with code 0
my word
Trace: my word
    at bar (c:\Users\User\OneDrive\수업\수업\node.js\v2.0\source\ch03\03_console3.js:3:1
5)
    at foo (c:\Users\User\OneDrive\수업\수업\node.js\v2.0\source\ch03\03_console3.js:5:5)
    at Object.<anonymous> (c:\Users\User\OneDrive\수업\수업\node.js\v2.0\source\ch03\03_c
onsole3.js:8:3)
    at Module._compile (internal/modules/cjs/loader.js:1137:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1157:10)
    at Module.load (internal/modules/cjs/loader.js:985:32)
    at Function.Module._load (internal/modules/cjs/loader.js:878:14)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:71:12)
    at internal/main/run_main_module.js:17:47
```

console

- console 예제 - 3

```
const errorMsg = 'the # is not even';
for (let number = 2; number <= 5; number += 1) {
    console.log('the # is ' + number);
    console.assert(number % 2 === 0, {number: number, errorMsg: errorMsg});
}
```

```
the # is 2
the # is 3
{number: 3, errorMsg: 'the # is not even'}
Assertion failed: [object Object]
the # is 4
the # is 5
```

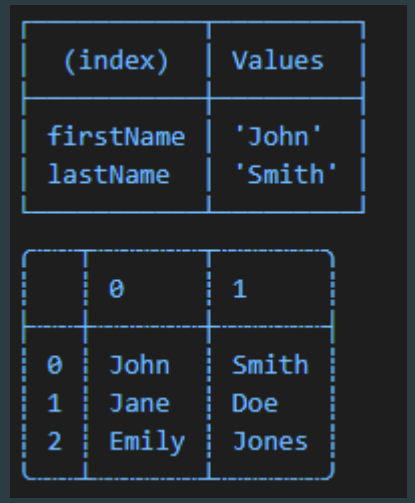
```
let user = "";

function greet() {
  console.count(user);
}

user = "bob";
greet();
user = "alice";
greet();
greet();
console.count("alice");
console.countReset("alice");
greet();
```

```
bob: 1
bob: 1
alice: 1
alice: 1
alice: 2
alice: 2
alice: 3
alice: 3
alice: 1
alice: 1
```

console

- console 예제 - 4

```
function Person(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
}

var me = new Person("John", "Smith");

console.table(me);

var people = [["John", "Smith"], ["Jane", "Doe"], ["Emily", "Jones"]]
console.table(people);
```



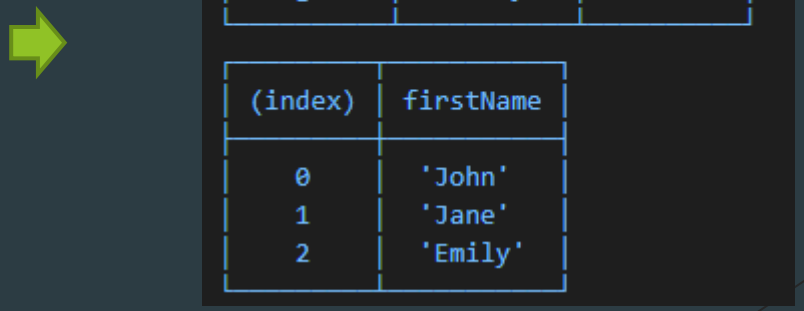
(index)	Values
firstName	'John'
lastName	'Smith'

	0	1
0	John	Smith
1	Jane	Doe
2	Emily	Jones

console

- console 예제 - 5

```
var family = {};

family.mother = new Person("Jane", "Smith");
family.father = new Person("John", "Smith");
family.daughter = new Person("Emily", "Smith");

console.table(family);

function Person(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
}

var john = new Person("John", "Smith");
var jane = new Person("Jane", "Doe");
var emily = new Person("Emily", "Jones");

console.table([john, jane, emily], ["firstName"]);
```



(index)	firstName	lastName
mother	'Jane'	'Smith'
father	'John'	'Smith'
daughter	'Emily'	'Smith'

(index)	firstName
0	'John'
1	'Jane'
2	'Emily'

__filename, __dirname

- __filename : 현재 파일의 경로 및 파일 이름을 출력한다.
- __dirname : 현재 파일의 경로를 출력한다.

```
console.log('filename : ', __filename);
console.log('dirname : ', __dirname);
```

```
filename :  c:\Users\User\OneDrive\수업\수업\node.js\v2.0\source\ch03\08_staticname.js
dirname :  c:\Users\User\OneDrive\수업\수업\node.js\v2.0\source\ch03
```

타이머 함수

- 타이머 함수 종류

함수	설명
setTimeout(callback, delay[, ...args])	일정 시간 후 함수를 실행한다.
setInterval(callback, delay[, ...args])	일정 시간 간격으로 함수를 반복 실행한다.
setImmediate(callback)	콜백 함수를 즉시 실행한다.
clearTimeout(timeoutObject)	실행되고 있는 timeout 을 중지한다.
clearInterval(intervalObject)	실행되고 있는 interval 을 중지한다.
clearImmediate(immediateObject)	실행되고 있는 immediateObject 를 중지한다.

타이머 함수

- 타이머 함수 예제

```
var num = 0;

var si = setInterval(function(){
    console.log("interval : ",num++);
}, 1000);

// clearInterval(interval);

var st = setTimeout(function(){
    clearInterval(si);
},6000);

clearTimeout(st);
```

```
interval :  0
interval :  1
interval :  2
interval :  3
interval :  4
interval :  5
interval :  6
interval :  7
interval :  8
interval :  9
interval :  10
```

TextEncoder, TextDecoder

- 암호화, 복호화 함수

클래스	설명
TextEncoder()	택스트를 인코딩한다.
TextDecoder([label], [options]);	텍스트를 디코딩한다. 디코딩할 시 label 에 따라 디코딩할 타입을 정할 수 있다.

```
let encoder = new TextEncoder("utf-8");

let e8Array = encoder.encode("Hello");
console.log(e8Array);

let d8Array = new Uint8Array([72, 101, 108, 108, 111]);
console.log( new TextDecoder().decode(d8Array) ); // Hello
```

```
Uint8Array(5) [72, 101, 108, 108, 111]
Hello
```

module

- module
  - 독립된 기능을 갖는 것(함수, 파일)들의 모임
  - 모듈은 Node.js에서 제공하는 것이 있고, 또는 누군가가 만들어 놓은 모듈이 있으며, 직접 만들 수도 있다.
  - 모듈은 크게 두가지로 나뉠 수 있다.
    - 외장 모듈
      - 일반 Node.js 개발자들이 만들어 놓은 모듈(라이브러리).
      - 외장 모듈을 사용하기 위해서는 npm(Node Package Manager)을 사용한다.
    - 내장 모듈
      - Node.js를 설치하고 나면 그 안에 이미 제공되어지는 모듈을 의미.
      - 내장 모듈은 이미 Node.js를 설치할 때 존재하기 때문에 npm을 사용하지 않는다.

module

- 모듈 사용을 위한 객체와 함수

객체/함수	설명
exports	해당 객체에 함수를 선언하여 외부에서 다른 파일에 위치한 함수를 사용할 수 있도록 해주는 객체.
require	module.exports 객체를 리턴하여 exports를 통해 선언한 함수를 사용할 수 있게 한다

```
// ext1.js

exports.f1 = function(){
    console.log("f1 함수 호출");
}
```

```
let ext1 = require('./ex/exp1.js');

ext1.f1();
```

```
f1 함수 호출
```

Buffer

- buffer 클래스
  - 버퍼 모듈은 이진 데이터의 스트림을 처리하는 방법을 제공한다.
  - 버퍼 개체는 Node.js의 글로벌 객체로, 필요 키워드를 사용하여 가져올 필요는 없다.
  - web에서 제공하는 버퍼와 동일한 역할을 한다.

Buffer

- buffer 함수

함수	설명
alloc()	지정된 길이의 버퍼 개체 생성
allocUnsafe()	지정된 길이의 0이 채워지지 않은 버퍼 생성
allocUnsafeSlow	지정된 길이의 0이 채워지지 않고 풀링되지 않은 버퍼 생성
byteLength()	지정된 개체의 바이트 수 반환
compare()	두 개의 버퍼 객체 비교
concat()	버퍼 객체의 배열을 하나의 버퍼 객체로 연결
copy()	버퍼 개체의 지정된 바이트 수 복사
entries()	버퍼 객체의 "색인" "바이트" 쌍의 엔트리 반환
equals()	두 개의 버퍼 객체를 비교하고 일치하면 true를 반환하고 그렇지 않으면 false
fill()	버퍼 객체를 지정된 값으로 채우기
from()	객체에서 버퍼 객체 생성(문자열/어레이/버퍼
includes()	버퍼 개체에 지정된 값이 포함되어 있는지 확인한다. 일치하는 항목이 있으면 true로 반환하고 그렇지 않으면 false
indexOf()	버퍼 개체에 지정된 값이 포함되어 있는지 확인한다. 없으면 -1
isBuffer()	객체가 버퍼 객체인지 확인
isEncoding()	버퍼 개체가 지정된 인코딩을 지원하는지 확인
keys()	버퍼 개체의 키 배열을 반환
lastIndexOf()	버퍼 개체에 지정된 값이 포함되어 있는지 끝에서부터 확인한다. 없으면 -1
length	버퍼 개체의 길이(바이트) 반환
poolSize	풀링에 사용되는 바이트 수를 설정하거나 반환
readDoubleBE()	버퍼 개체에서 64비트 더블 타입 읽기, 빅 엔디안으로 결과 반환
readDoubleLE()	버퍼 개체에서 64비트 더블 타입 읽기, 리틀 엔디안으로 결과 반환
readFloatBE()	버퍼 개체에서 32비트 float 타입을 읽고 결과를 빅 엔디안으로 반환
readFloatLE()	버퍼 객체에서 32비트 float 타입을 읽고 결과를 리틀 엔디안으로 반환
readInt8()	버퍼 개체에서 8비트 정수 읽기
readInt16BE()	버퍼 개체에서 16비트 정수 읽기, 빅 엔디안 으로 결과 반환
readInt16LE()	버퍼 개체에서 16비트 정수 읽기, 리틀 엔디안으로 결과 반환
readInt32BE()	버퍼 개체에서 32비트 정수 읽기, 빅 엔디안 으로 결과 반환
readInt32LE()	버퍼 개체에서 32비트 정수 읽기, 리틀 엔디안으로 결과 반환
readIntBE()	버퍼 개체에서 지정된 바이트 수 읽기, 빅 엔디안으로 결과 반환
readIntLE()	버퍼 개체에서 지정된 바이트 수 읽기, 리틀 엔디안으로 결과 반환
readUInt8()	버퍼 개체에서 서명되지 않은 8비트 정수 읽기
readUInt16BE()	버퍼 개체에서 서명되지 않은 16비트 정수 읽기, 빅 엔디안으로 결과 반환
readUInt16LE()	버퍼 개체에서 서명되지 않은 16비트 정수 읽기, 리틀 엔디안으로 결과 반환
readUInt32BE()	버퍼 개체에서 서명되지 않은 32비트 정수 읽기, 빅 엔디안으로 결과 반환
readUInt32LE()	버퍼 개체에서 서명되지 않은 32비트 정수 읽기, 리틀 엔디안으로 결과 반환
readUintBE()	버퍼 개체에서 지정된 바이트 수 읽기, 부호없는 정수로 결과를 반환함
readUIntLE()	버퍼 개체에서 지정된 바이트 수 읽기, 부호없는 정수로 결과를 반환함

Buffer

- buffer 함수

함수	설명
slice()	버퍼 객체를 지정된 위치에서 시작 및 종료하는 새 버퍼 객체로 슬라이스
swap16()	16비트 버퍼 개체의 바이트 순서 스왑
swap32()	32비트 버퍼 개체의 바이트 순서 스왑
swap64()	4비트 버퍼 개체의 바이트 순서 스왑
toString()	버퍼 개체의 문자열 버전 반환
toJSON()	버퍼 개체의 JSON 버전 반환
values()	버퍼 개체의 값 배열 반환
write()	지정된 문자열을 버퍼 개체에 쓰기
writeDoubleBE()	빅 엔디안을 사용하여 지정된 바이트를 버퍼 객체에 기록. 바이트는 64비트 2배여야 한다.
writeDoubleLE()	지정된 바이트를 리틀 엔디안을 사용하여 버퍼 객체에 기록. 바이트는 64비트 2배여야 한다.
writeFloatBE()	빅 엔디안을 사용하여 지정된 바이트를 버퍼 객체에 기록. 바이트는 32비트 부동이어야 한다.
writeFloatLE()	지정된 바이트를 리틀 엔디안을 사용하여 버퍼 객체에 기록. 바이트는 32비트 부동이어야 한다.
writeInt8()	지정한 바이트를 버퍼 개체에 쓰기. 바이트는 8비트 정수여야 함
writeInt16BE()	빅 엔디안을 사용하여 지정된 바이트를 버퍼 객체에 기록. 바이트는 16비트 정수여야 한다.
writeInt16LE()	지정된 바이트를 리틀 엔디안을 사용하여 버퍼 객체에 기록. 바이트는 16비트 정수여야 한다.
writeInt32BE()	빅 엔디안을 사용하여 지정된 바이트를 버퍼 객체에 기록. 바이트는 32비트 정수여야 한다.
writeInt32LE()	리틀 엔디안을 사용하여 지정된 바이트를 버퍼 객체에 기록. 바이트는 32비트 정수여야 한다.
writeIntBE()	빅 엔디안을 사용하여 지정된 바이트를 버퍼 객체에 기록.
writeIntLE()	지정된 바이트를 리틀 엔디안을 사용하여 버퍼 객체에 기록.
writeUInt8()	버퍼 개체에 지정된 바이트 쓰기. 바이트는 8비트 부호 없는 정수여야 함
writeUInt16BE()	빅 엔디안을 사용하여 지정된 바이트를 버퍼 개체에 기록. 바이트는 16비트 부호 없는 정수여야 한다.
writeUInt16LE()	리틀 엔디안을 사용하여 지정된 바이트를 버퍼 개체에 쓰기. 바이트는 16비트 부호 없는 정수여야 한다.
writeUInt32BE()	빅 엔디안을 사용하여 지정된 바이트를 버퍼 개체에 기록. 바이트는 32비트 부호 없는 정수여야 한다.
writeUInt32LE()	리틀 엔디안을 사용하여 지정된 바이트를 버퍼 개체에 쓰기. 바이트는 32비트 부호 없는 정수여야 한다.
writeUIntBE()	빅 엔디안을 사용하여 지정된 바이트를 버퍼 개체에 기록.
writeUIntLE()	리틀 엔디안을 사용하여 지정된 바이트를 버퍼 개체에 쓰기.

Buffer

- buffer 예제 - 1

```
var buf1 = Buffer.alloc(15);
console.log("1:",buf1);

var buf2 = Buffer.alloc(15, 'a');
console.log("2:",buf2);

var buf3 = Buffer.allocUnsafe(15);
console.log("3:",buf3);

buf3.fill(0);
console.log("4:",buf3);

var len = Buffer.byteLength(buf3);
console.log("len:",len);
```

```
1: Buffer(15) [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

2: Buffer(15) [97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97, 97]
3: Buffer(15) [0, 0, 0, 0, 0, 0, 0, 0, 192, 1, 18, 225, 131, 1, 0]
len: 15
```

Buffer

- buffer 예제 - 2

```
var buf1 = Buffer.from('abc');
var buf2 = Buffer.from('abc');
var x = Buffer.compare(buf1, buf2);
console.log(x); // 0

var buf1 = Buffer.from('a');
var buf2 = Buffer.from('b');
var x = Buffer.compare(buf1, buf2);
console.log(x); // -1

var buf1 = Buffer.from('b');
var buf2 = Buffer.from('a');
var x = Buffer.compare(buf1, buf2);
console.log(x); // 1
```

```
var buf1 = Buffer.from('a');
var buf2 = Buffer.from('b');
var buf3 = Buffer.from('c');
var arr = [buf1, buf2, buf3];

var buf = Buffer.concat(arr);
console.log(buf); // Buffer(3) [97, 98, 99]

var buf1 = Buffer.from('abcdefghijkl');
var buf2 = Buffer.from('HELLO');

buf2.copy(buf1, 2);
console.log(buf1.toString()); // abHELLOhijkl

buf2.copy(buf1, 2, 0, 2);
console.log(buf1.toString()); // abHEefghijkl

var buf = Buffer.from('abc');

for (x of buf.entries()) {
  console.log(x);
}
// [ 0, 97 ]
// [ 1, 98 ]
// [ 2, 99 ]

var buf1 = Buffer.from('abc');
var buf2 = Buffer.from('abc');

console.log(buf1.equals(buf2)); // true
```

```
var buf = Buffer.from('Hello, and welcome to Rome!');

console.log(buf.includes('welcome'));  // true
console.log(buf.indexOf('welcome'));  // 11
console.log(Buffer.isBuffer(buf));  // true
console.log(Buffer.isEncoding('utf8'));  // true
console.log(buf.lastIndexOf('e'));  // 25

var buf = Buffer.from('abc');

console.log(buf.toString()); // abc
for (x of buf.keys()) {
  console.log(x);
}
// 0,1,2
for (x of buf.values()) {
    console.log(x);
}
// 97,98,99
var x = Buffer.from('abcdef');
var y = x.slice(2,5);

console.log(y.toString());  // cde

var buf = Buffer.from('abcdef');
buf.write('qq',2);
console.log(buf.toString()); // abqqef
```