내장 객체

내장 객체

❖ 주요 내장 객체

- Number
- Boolean
- String
- Object
- Array
- Math
- Date
- RegExp

```
var str = '안녕하세요!';
// var str = new String('안녕하세요!');
console.log(str.length);
```

- 기본 데이터형에서는 new를 사용하지 않고 그냥 사용
  - 형 변환기 역할을 함

기본 자료형 객체

❖ String 객체

분류	멤버	개요
검색	indexOf(substr, [,start])	문자열 전방(start + 1번째 문자)부터 부분 문자열 substr을 검색
	lastIndexOf(substr, [,start])	문자열 후방(start + 1번째 문자)부터 부분 문자열 substr을 검색
	startsWith(search [,pos]) ES2015	문자열이 지정된 부분 문자열 search로 시작하는가(인수 pos는 검색 시작 위치)
	endsWith(search [,pos]) ES2015	문자열이 지정된 부분 문자열 search로 종료하는가
	includes (search [,pos]) ES2015	문자열이 지정된 부분 문자열 search를 포함하는가
부분 문자열	charAt(n)	n + 1번째의 문자를 추출
	slice(start [,end])	문자열부터 start + 1~end번째 문자를 추출
	substring(start [,end])	문자열부터 start + 1~end번째 문자를 추출
	substr(start [,cnt])	문자열부터 start + 1번째 문자부터 cnt 수만큼의 문자를 추출
	split(str [,limit])	문자열을 분할 문자열 str로 분할하여 그 결과를 배열로 취득(인수 limit는 최대 분할수)

기본 자료형 객체

❖ String 객체

분류	멤버	개요
정규 표현	match(reg)	정규 표현 reg로 문자열을 검색, 일치한 부분 문자열을 취득
	replace(reg, rep)	정규 표현 reg로 문자열을 검색, 일치한 부분을 부분 문자열 rep로 치환
	search(reg)	정규 표현 reg로 문자열을 검색, 일치한 맨 처음 문자 위치를 취득
대문자⇔ 소문자	toLowerCase()	소문자로 치환
	toUpperCase()	대문자로 치환
코드 변환	charCodeAt(n)	n + 1번째의 문자를 Latin-1 코드로 변환
	*fromCharCode(c1, c2…)	Latin-1 코드 c1, c2, …를 문자로 변환
	codePointAt(n) ES2015	n + 1번째의 문자를 UTF-16 인코딩된 코드 포인트값으로 변환
	*fromCodePoint(num,…) ES2015	코드 포인트값으로부터 문자열을 생성
그 외	concat(str)	문자열 뒤쪽에 문자열 str을 연결
	repeat(num) ES2015	문자열을 num 숫자만큼 반복한 것을 취득
	trim()	문자열의 전후에서 공백을 삭제
	length	문자열의 길이를 취득

기본 자료형 객체

❖ String

```
var str1 = '뜰에 뜰에 뜰에는 닭이 있다.';

console.log(str1.indexOf('뜰'));
console.log(str1.lastIndexOf('뜰'));
console.log(str1.indexOf('뜰', 3));
console.log(str1.lastIndexOf('에', 5));
console.log(str1.indexOf('가든'));
console.log(str1.startsWith('뜰'));
console.log(str1.endsWith('뜰'));
console.log(str1.includes('뜰'));
```

기본 자료형 객체

❖ String

```
var str2 = 'Wings프로젝트';
var str3 = '💩싸서';
var str4 = '   wings   ';

console.log(str2.charAt(4));
console.log(str2.slice(5, 8));
console.log(str2.substring(5, 8));
console.log(str2.substr(5, 3));
console.log(str2.split('s'));
console.log(str1.split('에', 3));
console.log(str2.charCodeAt(0));
console.log(String.fromCharCode(87, 105, 110, 103));
console.log(str3.codePointAt(0));
console.log(String.fromCodePoint(128169));
console.log(str2.concat(' 유한회사'));
console.log(str2.repeat(2));
console.log(str4.trim());
console.log(str2.length);
```

기본 자료형 객체

❖ Number 객체

메소드	toString(rad)	rad 진수값으로 변환(rad는 2~36)
	toExponential(dec)	지수 형식으로 변환(dec는 소수점 이하의 행수)
	toFixed(dec)	소수점 이하의 행수 dec 반올림
	toPrecision(dec)	지정 행수로 변환(행수가 부족한 경우는 0으로 보충)
	*isNaN(num) ES2015	NaN(Not a Number)인지를 판정
	*isFinite(num) ES2015	유한값인지를 판정
	*isInteger(num) ES2015	정수값인지를 판정
	*isSafeInteger(num) ES2015	Safe Integer인지 (올바르게 IEEE-754배 정도 수로 표현할 수 있는가)를 판정
	*parseFloat(str) ES2015	문자열을 소수점수로 변환
	*parseInt(str [,radix]) ES2015	문자열을 정수로 변환(인수 radix는 기수)

기본 자료형 객체

❖ Number 객체

```
var num1 = 255;
console.log(num1.toString(16));
console.log(num1.toString(8));

var num2 = 123.45678;
console.log(num2.toExponential(2));
console.log(num2.toFixed(3));
console.log(num2.toFixed(7));
console.log(num2.toPrecision(10));
console.log(num2.toPrecision(6));
```

기본 자료형 객체

❖ 문자열을 숫자로 변환하기

```
var n = '123xxx';
console.log(Number(n));
console.log(Number.parseFloat(n));
console.log(Number.parseInt(n));

var h = '0x10';
console.log(Number(h));
console.log(Number.parseFloat(h));
console.log(Number.parseInt(h));

var b = '0b11';
console.log(Number(b));
console.log(Number.parseFloat(b));
console.log(Number.parseInt(b));

var e = '1.01e+2';
console.log(Number(e));
console.log(Number.parseFloat(e));
console.log(Number.parseInt(e));
```

기본 자료형 객체

❖ 산술 연산자에 의한 문자열 처리

- + : 문자열로 해석
- 그 외 : 숫자로 해석

```
console.log(typeof(123 + ''));
console.log(typeof('123' - 0));
```

기본 자료형 객체

❖ Math 객체

분류	멤버	개요
기본	abs(num)	절대치
	clz32(num) ES2015	32비트 바이너리로 표현했을 때 앞부분에 채워진 0의 개수
	max(num1, num2)	num1, num2 중에서 큰 쪽의 값
	min(num1, num2)	num1, num2 중에서 작은 쪽의 값
	pow(base, p)	거듭제곱(base값의 p승)
	random()	0~1미만의 난수
	sign(num) ES2015	지정한 값이 양수인 경우는 1, 음수의 경우는 -1, 0인 경우는 0
자리올림/자리버림	ceil(num)	소수점 이하 올림(num 이상의 최소 정수)
	floor(num)	소수점 이하 버림(num이하의 최대 정수)
	round(num)	반올림

기본 자료형 객체

❖ Math 객체

기본	abs(num)	절대치
	clz32(num) ES2015	32비트 바이너리로 표현했을 때 앞부분에 채워진 0의 개수
	max(num1, num2)	num1, num2 중에서 큰 쪽의 값
	min(num1, num2)	num1, num2 중에서 작은 쪽의 값
	pow(base, p)	거듭제곱(base값의 p승)
	random()	0~1미만의 난수
	sign(num) ES2015	지정한 값이 양수인 경우는 1, 음수의 경우는 -1, 0인 경우는 0
자리올림/자리버림	ceil(num)	소수점 이하 올림(num 이상의 최소 정수)
	floor(num)	소수점 이하 버림(num이하의 최대 정수)
	round(num)	반올림
	trunc(num) ES2015	소수 부분을 단순히 버림(정수 부분을 취득)
제곱근	*SQRT1_2	1/2의 제곱근
	*SQRT2	2의 제곱근
	sqrt(num)	제곱근
	cbrt(num) ES2015	세제곱근
	hypot(x1, x2, …) ES2015	인수의 제곱합의 제곱근

기본 자료형 객체

❖ Math 객체

삼각함수	*PI	원주율
	cos(num)	코사인
	sin(num)	사인
	tan(num)	탄젠트
	acos(num)	아크 코사인
	asin(num)	아크 사인
	atan(num)	아크 탄젠트
	atan2(y, x)	2변수의 아크 탄젠트
	cosh(num) ES2015	쌍곡 코사인
	sinh(num) ES2015	쌍곡 사인
	tanh(num) ES2015	쌍곡 탄젠트
	acosh(num) ES2015	역쌍곡 코사인
	asinh(num) ES2015	역쌍곡 사인
	atanh(num) ES2015	역쌍곡 탄젠트

기본 자료형 객체

❖ Math 객체

로그/지수함수	*E	자연 로그의 밑에 해당하는 수학 상수. 2.718281828459045
	*LN2	2의 자연 로그 값 0.6931471805599453
	*LN10	10의 자연 로그 값. 2.302585092994046
	*LOG2E	2을 밑으로 한 e의 로그. 1.4426950408889634
	*LOG10E	10을 밑으로 한 e의 로그. 0.43429448190325182
	log(*num*)	자연 로그
	log10(*num*) ES2015	밑을 10으로 하는 로그
	log2(*num*) ES2015	밑을 2으로 하는 로그
	log1p(*num*) ES2015	인수 x에 1을 더한 것의 자연 로그
	exp(*num*)	지수 함수(e의 거듭제곱)
	expm1(*num*) ES2015	$e^{num}-1$

값의 집합을 처리/조작하기

❖ Array

```
var ary = ['배트맨', '슈퍼맨', '아쿠아맨'];
```

```
var ary = new Array('배트맨', '슈퍼맨', '아쿠아맨');
var ary = new Array();
var ary = new Array(10);
```

값의 집합을 처리/조작하기

❖ Array 주요 멤버

- 원본이 변하는 경우 *로 표시

기본	length	배열의 크기
	isArray(obj)	지정한 객체가 배열인가(정적 메소드)
	toString()	'요소, 요소, …'의 형식으로 문자열 변환
	toLocaleString()	배열을 문자열로 변환(구분 문자는 로케일에 따라 변화)
	indexOf(elm [,index])	지정한 요소와 일치한 첫 요소의 키를 취득(index는 검색 시작 위치)
	lastIndexOf(elm [,index])	지정한 요소와 일치한 마지막 요소의 키를 취득(index는 검색 시작 위치)
	entries() ES2015	모든 키/값을 취득
	keys() ES2015	모든 키를 취득
	values() ES2015	모든 값을 취득

값의 집합을 처리/조작하기

❖ Array 주요 멤버

가공	concat(ary)	지정 배열을 현재의 배열에 연결
	join(del)	배열 내의 요소를 구분 문자 del로 연결
	slice(start [,end])	start + 1 ~ end번째의 요소를 빼냄
	*splice(start, cnt [,rep [,…]])	배열 내의 start + 1 ~ start + cnt번째의 요소를 rep, …로 치환
	from(alike [,map [,othis]) ES2015	배열과 비슷한 종류의 객체와 열거 가능한 객체를 배열로 변환(정적 메소드, 249쪽을 참조)
	of(e1, …) ES2015	가변길이 인수를 배열에 변환(정적 메소드)
	*copyWithin(target, start [,end]) ES2015	start + 1 ~ end번째의 요소를 target + 1번째 위치부터 복사(요소 수는 원래와 변환 없음)
	*fill(var [,start [,end]]) ES2015	배열 내의 start + 1 ~ end번째의 요소를 var로 치환
추가/삭제	*pop()	배열 끝의 요소를 취득하여 삭제
	*push(data)	배열 끝에 요소를 추가
	*shift()	배열 선두의 요소를 취득하여 삭제
	*unshift(data1 [,data2,…])	배열 선두에 지정 요소를 추가

값의 집합을 처리/조작하기

❖ Array 주요 멤버

정렬	*reverse()	역순으로 정렬(반전)
	*sort([fnc])	요소를 오름차순으로 정렬
콜백	forEach(fnc [,that])	배열 내의 요소를 함수로 순서대로 처리
	map(fnc [,that])	배열 내의 요소를 함수로 순서대로 가공
	every (fnc [,that])	모든 배열 내의 요소가 조건 fnc에 일치하는가
	some(fnc [,that])	일부 배열 내의 요소가 조건 fnc에 일치하는가
	filter(fnc [,that])	조건 fnc에 일치한 요소로 배열을 생성
	find(fnc [,that]) ES2015	함수 fnc가 처음 true를 반환한 요소를 취득
	findIndex(fnc [,that]) ES2015	함수 fnc가 처음 true를 반환한 요소의 인덱스 번호를 취득
	reduce (fnc [,init])	바로 옆의 두 요소를 왼쪽부터 오른쪽으로 함수 fnc로 처리하여 단일 값으로 한다(인수 init는 초깃값)
	reduceRight (fnc [,init])	바로 옆의 두 요소를 오른쪽부터 왼쪽으로 함수 fnc로 처리하여 단일 값으로 한다(인수 init는 초깃값)

값의 집합을 처리/조작하기

```
var data = [];
data.push(1);
data.push(2);
data.push(3);
console.log(data.pop());
console.log(data.pop());
console.log(data.pop());
```

```
var data = [];
data.push(1);
data.push(2);
data.push(3);
console.log(data.shift());
console.log(data.shift());
console.log(data.shift());
```

값의 집합을 처리/조작하기

❖ 배열 요소의 추가/치환/삭제 - splice

```
array.splice(index, many [,elem1 [,elem2, …]])
```

array: 배열 객체　　*index*: 요소의 추출 시작 위치

many: 추출 요소 수　　*elem1*, *elem2*…: 삭제 부분에 삽입할 요소

```
var data = ['A', 'B', 'C', 'D', 'E'];
console.log(data.splice(3, 2, 'F', 'G'));
console.log(data);
console.log(data.splice(3, 2));
console.log(data);
console.log(data.splice(1, 0, 'H'));
console.log(data);
```

값의 집합을 처리/조작하기

❖ 배열의 내용을 순서대로 처리하기 - forEach

```
array.forEach(callback [,that])
        array: 배열 객체
        callback: 개별 요소를 처리하기 위한 함수
        that: 함수 callback 안에서 this(5.1.5절)가 나타내는 객체
```

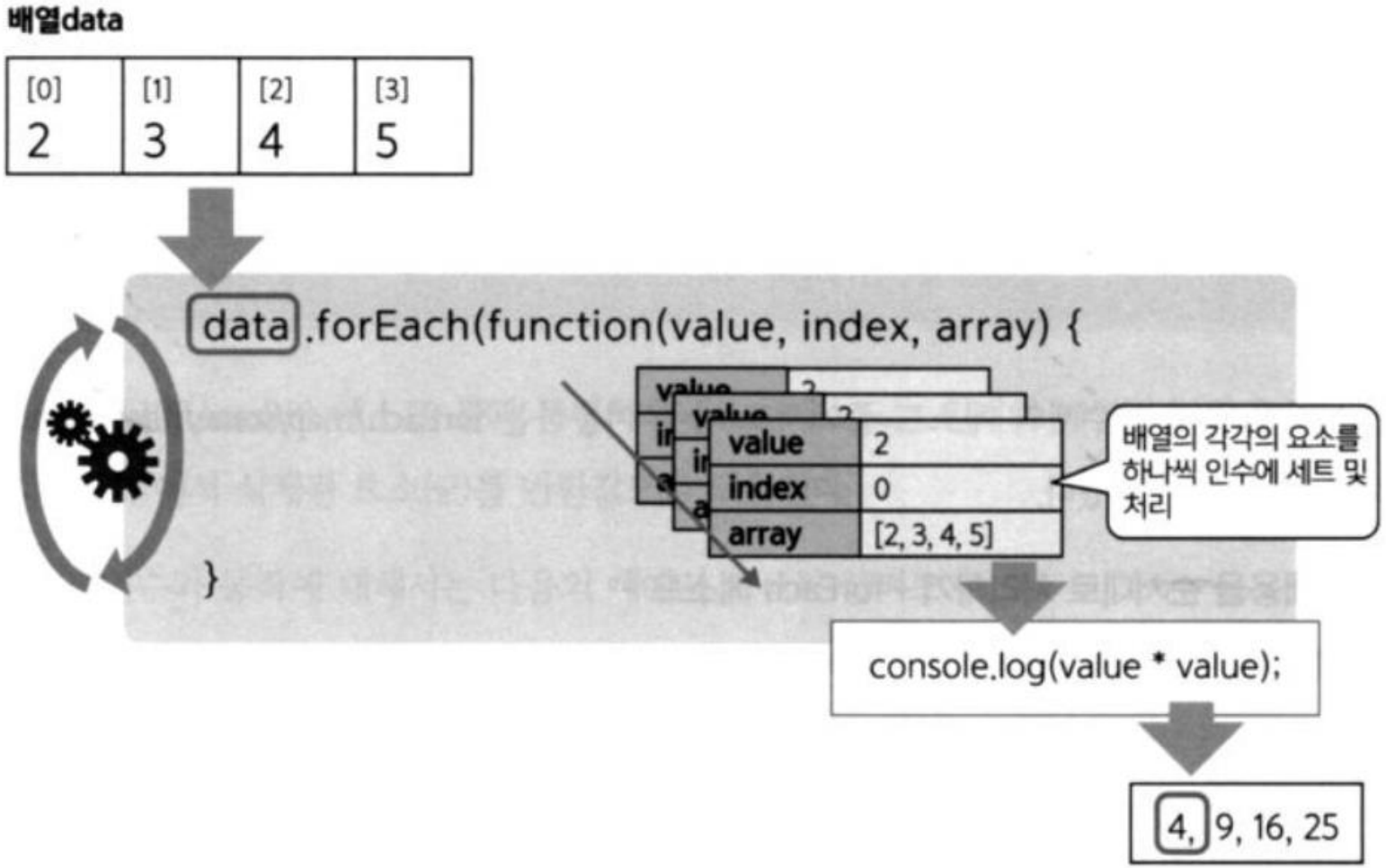
```
var data = [2, 3, 4, 5];

data.forEach(function(value, index, array) {
  console.log(value * value);
});
```

- 콜백 함수의 인자
  - 요소의 값
  - 요소의 인덱스
  - 전체 배열

값의 집합을 처리/조작하기

❖ 배열의 내용을 순서대로 처리하기 - forEach



값의 집합을 처리/조작하기

❖ 배열을 지정된 규칙으로 가공하기 - map

```
array.map(callback [,that])
        array: 배열 객체
        callback: 개별 요소를 처리하기 위한 함수
        that: 함수 callback 안에서 this(5.1.5절)가 나타내는 객체
```

```
var data = [2, 3, 4, 5];
var result = data.map(function(value, index, array) {
  return value * value;
});

console.log(result);
```

값의 집합을 처리/조작하기

❖ 배열에 조건이 일치하는 요소가 존재하는지 확인하기 - some

```
array.some(callback [,that])
        array 배열 객체
        callback: 개별 요소를 처리하기 위한 함수
        that: 함수 callback 안에서 this(5.1.5절)가 나타내는 객체
```

```
var data = [4, 9, 16, 25];
var result = data.some(function(value, index, array) {
  return value % 3 === 0;
});

if (result) {
  console.log('3의 배수가 발견되었습니다.');
} else {
  console.log('3의 배수를 찾을 수 없었습니다.');
}
```

값의 집합을 처리/조작하기

❖ 배열의 내용을 특정의 조건으로 필터링 - filter

```
array.filter(callback [,that])
```

array: 배열 객체

callback: 개별 요소를 처리하기 위한 함수

that: 함수 callback 안에서 this(5.1.5절)가 나타내는 객체

```
var data = [4, 9, 16, 25];
var result = data.filter(function(value, index, array) {
  return value % 2 === 1;
});

console.log(result);
```

값의 집합을 처리/조작하기

❖ 정렬하기 - sort

- 정렬함수
  - 인자로 전달된 두 개의 값을 비교하여 부호를 리턴

```
var ary = [5, 25, 10];

console.log(ary.sort());  //  문자열로 정렬

console.log(ary.sort(function(x, y) {
  return x - y;
}));
```

값의 집합을 처리/조작하기

❖ 정렬하기 - sort

```
var classes = ['부장', '과장', '주임', '담당'];
var members = [
  { name: '남상미', clazz: '주임' },
  { name: '김준수', clazz: '부장' },
  { name: '정인식', clazz: '담당' },
  { name: '남궁민', clazz: '과장' },
  { name: '이상주', clazz: '담당' },
];
console.log(members.sort(function(x, y) {
  return classes.indexOf(x.clazz) - classes.indexOf(y.clazz);
}))
```

날짜/시간 데이터 조작

❖ **Date 객체 주요 메소드**

로컬 (취득)	getFullYear()	년(4자리 수)
	getMonth()	월(0~11)
	getDate()	일(1~31)
	getDay()	요일(0: 일요일~6: 토요일)
	getHours()	시(0~23)
	getMinutes()	분(0~59)
	getSeconds()	초(0~59)
	getMilliseconds()	밀리 세컨드(0~999)
	getTime()	1970/01/01 00:00:00로부터 경과 밀리 세컨드
	getTimezoneOffset()	협정세계시와의 시차

날짜/시간 데이터 조작

❖ **Date 객체 주요 메소드**

로컬 (설정)	setFullYear(y)	년(4자리 수)
	setMonth(m)	월(0~11)
	setDate(d)	일(1~31)
	setHours(h)	시(0~23)
	setMinutes(m)	분(0~59)
	setSeconds(s)	초(0~59)
	setMilliseconds(ms)	밀리 세컨드(0~999)
	setTime(ts)	1970/01/01 00:00:00로부터 경과 밀리 세컨드

날짜/시간 데이터 조작

❖ Date 객체 주요 메소드

협정시 (취득)	getUTCFullYear()	년(4자리 수)
	getUTCMonth()	월(0~11)
	getUTCDate()	일(1~31)
	getUTCDay()	요일(0: 일요일~6: 토요일)
	getUTCHours()	시(0~23)
	getUTCMinutes()	분(0~59)
	getUTCSeconds()	초(0~59)
	getUTCMilliseconds()	밀리 세컨드(0~999)
협정시 (설정)	setUTCFullYear(y)	년(4자리 수)
	setUTCMonth(m)	월(0~11)
	setUTCDate(d)	일(1~31)

날짜/시간 데이터 조작

❖ **Date 객체 주요 메소드**

협정시 (취득)	getUTCFullYear()	년(4자리 수)
	getUTCMonth()	월(0~11)
	getUTCDate()	일(1~31)
	getUTCDay()	요일(0: 일요일~6: 토요일)
	getUTCHours()	시(0~23)
	getUTCMinutes()	분(0~59)
	getUTCSeconds()	초(0~59)
	getUTCMilliseconds()	밀리 세컨드(0~999)
협정시 (설정)	setUTCFullYear(y)	년(4자리 수)
	setUTCMonth(m)	월(0~11)
	setUTCDate(d)	일(1~31)

❖ Date 객체 주요 메소드

- o *는 static 메서드

해석	* parse(dat)	날짜 문자열을 해석해 1970/01/01 00:00:00로부터 경과 밀리 세컨드를 취득
	* UTC(y, m, d [,h [,mm [,s[,ms]]]])	날짜 정보를 기초로 1970/01/01 00:00:00로부터 경과 밀리 세컨드를 취득(협정시)
	* now()	협정세계시에서의 현재 날짜를 1970/01/01 00:00:00로부터 경과 밀리초로 취득
문자열 변환	toUTCString()	협정세계시를 문자열로 취득
	toLocaleString()	로컬시를 문자열로 취득
	toDateString()	일자 부분을 문자열로 취득
	toTimeString()	시각 부분을 문자열로 취득
	toLocaleDateString()	지역 정보에 따라서 날짜 부분을 문자열로 취득
	toLocaleTimeString()	지역 정보에 따라서 시각 부분을 문자열로 취득
	toString()	일시를 문자열로 취득
	toJSON()	일시를 JSON 문자열(3.7.3절)로 취득

날짜/시간 데이터 조작

❖ Date 객체

```
var dat = new Date(2016, 11, 25, 11, 37, 15, 999);
console.log(dat);
console.log(dat.getFullYear());
console.log(dat.getMonth());
console.log(dat.getDate());
console.log(dat.getDay());
console.log(dat.getHours());
console.log(dat.getMinutes());
console.log(dat.getSeconds());
console.log(dat.getMilliseconds());
console.log(dat.getTime());
console.log(dat.getTimezoneOffset());

console.log(dat.getUTCFullYear());
console.log(dat.getUTCMonth());
console.log(dat.getUTCDate());
console.log(dat.getUTCDay());
console.log(dat.getUTCHours());
console.log(dat.getUTCMinutes());
console.log(dat.getUTCSeconds());
console.log(dat.getUTCMilliseconds());
```

날짜/시간 데이터 조작

❖ Date 객체

```
var dat2 = new Date();
dat2.setFullYear(2017);
dat2.setMonth(7);
dat2.setDate(5);
dat2.setHours(11);
dat2.setMinutes(37);
dat2.setSeconds(15);
dat2.setMilliseconds(513);

console.log(dat2.toLocaleString());
console.log(dat2.toUTCString());
console.log(dat2.toDateString());
console.log(dat2.toTimeString());
console.log(dat2.toLocaleDateString());
console.log(dat2.toLocaleTimeString());
console.log(dat2.toJSON());

console.log(Date.parse('2016/11/05'));
console.log(Date.UTC(2016, 11, 5));
console.log(Date.now());
```