JavaScript

Math – 김근형 강사

Math

- Math 속성

속성	설명
Math.E	오일러의 상수이며 자연로그의 밑. 약 2.718.
Math.LN2	2의 자연로그. 약 0.693.
Math.LN10	10의 자연로그. 약 2.303.
Math.LOG2E	밑이 2인 로그 E. 약 1.443.
Math.LOG10E	밑이 10인 로그 E. 약 0.434.
Math.PI	원의 둘레와 지름의 비율. 약 3.14159.
Math.SQRT1_2	1/2의 제곱근. 약 0.707.
Math.SQRT2	2의 제곱근. 약 1.414.

Math

- Math 속성

```
console.log(Math.E);
console.log(Math.LN2);
console.log(Math.LN10);
console.log(Math.LOG2E);
console.log(Math.LOG10E);
console.log(Math.PI);
console.log(Math.SQRT1_2);
console.log(Math.SQRT2);
```

```
2.718281828459045
0.6931471805599453
2.302585092994046
1.4426950408889634
0.4342944819032518
3.141592653589793
0.7071067811865476
1.4142135623730951
```

Math

- Math 기능
  - Math에서 지원하는 기능은 다음과 같다.

함수명	설명
abs()	인수의 절댓값(absolute value)을 반환한다. 절댓값은 반드시 0 또는 양수이어야 한다.
round()	인수의 소수점 이하를 반올림한 정수를 반환한다.
ceil()	인수의 소수점 이하를 올림한 정수를 반환한다.
floor()	인수의 소수점 이하를 내림한 정수를 반환한다. Math.ceil의 반대 개념이다. 양수인 경우, 소수점 이하를 떼어 버린 다음 정수를 반환한다. 음수인 경우, 소수점 이하를 떼어 버린 다음 -1을 한 정수를 반환한다
sqrt()	인수의 제곱근을 반환한다.
random()	임의의 부동 소수점을 반환한다. 반환된 부동 소수점은 0부터 1 미만이다. 즉, 0은 포함되지만 1은 포함되지 않는다.
pow()	첫번째 인수를 밑(base), 두번째 인수를 지수(exponent)로하여 거듭제곱을 반환한다.
max()	인수 중에서 가장 큰 수를 반환한다.
min()	인수 중에서 가장 작은 수를 반환한다.

Math

- abs() : 절대값

```
console.log(Math.abs(-1));          // 1
console.log(Math.abs('-1'));        // 1
console.log(Math.abs(''));          // 0
console.log(Math.abs([]));          // 0
console.log(Math.abs(null));        // 0
console.log(Math.abs(undefined));// NaN
console.log(Math.abs({}));          // NaN
console.log(Math.abs('string')); // NaN
console.log(Math.abs());            // NaN
```

- round() : 반올림

```
console.log(Math.round(1.4));  // 1
console.log(Math.round(1.6));  // 2
console.log(Math.round(-1.4)); // -1
console.log(Math.round(-1.6)); // -2
console.log(Math.round(1));    // 1
console.log(Math.round());     // NaN
```

Math

- ceil() : 올림

```
console.log(Math.ceil(1.4));  // 2
console.log(Math.ceil(1.6));  // 2
console.log(Math.ceil(-1.4)); // -1
console.log(Math.ceil(-1.6)); // -1
console.log(Math.ceil(1));    // 1
console.log(Math.ceil());     // NaN
```

- floor() : 버림

```
console.log(Math.floor(1.9));  // 1
console.log(Math.floor(9.1));  // 9
console.log(Math.floor(-1.9)); // -2
console.log(Math.floor(-9.1)); // -10
console.log(Math.floor(1));    // 1
console.log(Math.floor());     // NaN
```

Math

- sqrt() : 제곱근

```
console.log(Math.sqrt(9));  // 3
console.log(Math.sqrt(-9)); // NaN
console.log(Math.sqrt(2));  // 1.414213562373095
console.log(Math.sqrt(1));  // 1
console.log(Math.sqrt(0));  // 0
console.log(Math.sqrt());   // NaN
```

- random() : 랜덤값

```
console.log(Math.random()); // 0 ~ 1 미만의 부동 소수점 (0.8208720231391746)

// 1 ~ 10의 랜덤 정수 취득
// 1) Math.random로 0 ~ 1 미만의 부동 소수점을 구한 다음, 10을 곱해 0 ~ 10 미만의 부동 소수점을 구한다.
// 2) 0 ~ 10 미만의 부동 소수점에 1을 더해 1 ~ 10까지의 부동 소수점을 구한다.
// 3) Math.floor으로 1 ~ 10까지의 부동 소수점의 소수점 이하를 떼어 버린 다음 정수를 반환한다.
const random = Math.floor((Math.random() * 10) + 1);
console.log(random); // 1 ~ 10까지의 정수
```

Math

- pow() : 거듭제곱

```
console.log(Math.pow(2, 8));  // 256
console.log(Math.pow(2, -1)); // 0.5
console.log(Math.pow(2));     // NaN

// ES7(ECMAScript 2016) Exponentiation operator(거듭 제곱 연산자)
console.log(2 ** 8); // 256
```

- max() : 최대값

```
console.log(Math.max(1, 2, 3)); // 3

// 배열 요소 중에서 최대값 취득
const arr = [1, 2, 3];
const max = Math.max.apply(null, arr);
console.log(max); // 3

// ES6 Spread operator
Math.max(...arr); // 3
```

Math

- min() : 최소값

```
console.log(Math.min(1, 2, 3)); // 1

// 배열 요소 중에서 최소값 취득
const arr = [1, 2, 3];
const min = Math.min.apply(null, arr);
console.log(min); // 1

// ES6 Spread operator
console.log(Math.min(...arr)); // 1
```

Math

- Math
  - Math 오브젝트에 추가된 상수는 없으며 함수가 추가되었다. 추가된 함수 목록은 아래에 정의하였다.

함수명	설명	함수명	설명
sinh()	쌍곡 사인	log2()	2를 밑으로 한 로그 값
asinh()	쌍곡 아크사인	expm1()	자연로그 상수(e) 의 x승 – 1
cosh()	쌍곡 코사인	hypot()	제곱근
acosh()	쌍곡 아크코사인	cbrt()	세제곱근
tanh()	쌍곡 탄젠트	sign()	사인(sign)값
atanh()	쌍곡 아크탄젠트	trunc()	소수를 제외한 정수(버림)
log1p()	log(1+파라미터 값)	imul()	파라미터 값을 곱하고 결과를 32비트로 변환
log10()	10을 밑으로 한 로그 값	clz32()	32비트 값에서 0비트 수
fround()	32비트 유동 소수 값		

Math

- sinh() : 쌍곡 사인

```
console.log(Math.sinh(0));
console.log(Math.sinh(1));
```

```
0
1.1752011936438014
```

- asinh() : 쌍곡 아크사인

```
console.log(Math.asinh(0));
console.log(Math.asinh(1));
```

```
0
0.881373587019543
```

- cosh() : 쌍곡 코사인

```
console.log("1:", Math.cosh(0));
console.log("2:", Math.cosh(1));
console.log("3:", Math.cosh(-1));
```

```
1: 1
2: 1.5430806348152437
3: 1.5430806348152437
```

Math

- acosh() : 쌍곡 아크코사인

```
console.log("1:", Math.acosh(0));
console.log("2:", Math.acosh(1));
console.log("3:", Math.acosh(2));
```

```
1: NaN
2: 0
3: 1.3169578969248166
```

- tanh() : 쌍곡 탄젠트

```
console.log("1:", Math.tanh(0));
console.log("2:", Math.tanh(1));
console.log("3:", Math.tanh(Infinity));
```

```
1: 0
2: 0.7615941559557649
3: 1
```

- atanh() : 쌍곡 아크탄젠트

```
console.log("1:", Math.atanh(-2));
console.log("2:", Math.atanh(2));
console.log("3:", Math.atanh(-1));

console.log("4:", Math.atanh(1));
console.log("5:", Math.atanh(0));
console.log("6:", Math.atanh(0.5));
```

```
1: NaN
2: NaN
3: -Infinity
4: Infinity
5: 0
6: 0.5493061443340548
```

Math

- log2()

```
console.log("1:", Math.log2(3));
console.log("2:", Math.log2(2));
console.log("3:", Math.log2(1));
console.log("4:", Math.log2(0));
console.log("5:", Math.log2(-1));
```

```
1: 1.584962500721156
2: 1
3: 0
4: -Infinity
5: NaN
```

- log10()

```
console.log("1:", Math.log10(100));
console.log("2:", Math.log10(10));
console.log("3:", Math.log10(2));
console.log("4:", Math.log10(1));
console.log("5:", Math.log10(-1));
```

```
1: 2
2: 1
3: 0.3010299956639812
4: 0
5: NaN
```

Math

- log1p()

```
console.log("1:", Math.log1p(1));
console.log("2:", Math.log1p(0));
console.log("3:", Math.log1p(-1));
console.log("4:", Math.log1p(-2));
```

```
1: 0.6931471805599453
2: 0
3: -Infinity
4: NaN
```

- expm1()

```
console.log("1:", Math.expm1(-1));
console.log("2:", Math.expm1(0));
console.log("3:", Math.expm1(1));

console.log("4:", Math.expm1(Infinity));
console.log("5:", Math.expm1(-Infinity));
```

```
1: -0.6321205588285577
2: 0
3: 1.718281828459045
4: Infinity
5: -1
```

Math

- hypot()

$$\text{Math.hypot}(v_1, v_2, \cdots, v_n) = \sqrt{\sum_{i=1}^{n} v_i^2} = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$$

```
console.log("1:", Math.hypot(3));
console.log("2:", Math.hypot(3, 4));
console.log("3:", Math.hypot(3, 4, 5));

console.log("4:", Math.hypot());
console.log("5:", Math.hypot(-3, -4));
```

```
1: 3
2: 5
3: 7.0710678118654755
4: 0
5: 5
```

- cbrt()

```
console.log("1:", Math.cbrt());
console.log("2:", Math.cbrt(0));

console.log("3:", Math.cbrt(1));
console.log("4:", Math.cbrt(3));
console.log("5:", Math.cbrt(8));
```

```
1: NaN
2: 0
3: 1
4: 1.4422495703074083
5: 2
```

Math

- sign() : NaN -> NaN / ±0 → ±0 / ±숫자 → ±1

```
console.log("1:", Math.sign(NaN));
console.log("2:", Math.sign(0));
console.log("3:", Math.sign(-0));

console.log("4:", Math.sign(1));
console.log("5:", Math.sign(-1));
```

```
1: NaN
2: 0
3: -0
4: 1
5: -1
```

- trunc()

```
console.log("1:", Math.trunc(12.78));
console.log("2:", Math.trunc(0.12));

console.log("3:", Math.trunc(-0.12));
console.log("4:", Math.trunc(-1.23));

console.log("5:", Math.trunc());
console.log("6:", Math.trunc(NaN));
```

```
1: 12
2: 0
3: -0
4: -1
5: NaN
6: NaN
```

Math

- imul()

```
// 2의 32승 값은 4,294,967,296
console.log("1:", Math.imul(2, 4));
console.log("2:", Math.imul(-3, -4));

console.log("3:", Math.imul(123456, 1000));
console.log("4:", Math.imul(123456, 10000));
console.log("5:", Math.imul(123456, 100000));
```

```
1: 8
2: 12
3: 123456000
4: 1234560000
5: -539301888
```

- clz32()

```
// 31
console.log("1:", Math.clz32(1));
// 30: 000000~0000010이 되므로
console.log("2:", Math.clz32(2));
// 32: 값이 0으로 처리되므로
console.log("3:", Math.clz32());

// 30: 소수 값은 무시되므로
console.log("4:", Math.clz32(2.5));
// 1로 변환되므로
console.log("5:", Math.clz32(true));
```

```
1: 31
2: 30
3: 32
4: 30
5: 31
```

Math

- fround()

```
let value = 0.1 + 0.2;
console.log("1:", value);
console.log("2:", Math.fround(value));

console.log("3:", 1.23);
console.log("4:", Math.fround(1.23));
```

```
1: 0.30000000000000004
2: 0.30000001192092896
3: 1.23
4: 1.2300000190734863
```