

6.13 Math object

Introduction to the Math object

The **Math** object provides properties for mathematical constants and methods to perform mathematical functions.

PARTICIPATION ACTIVITY

6.13.1: Math properties.



Match the Math property to the property's description.

If unable to drag and drop, refresh the page.

Math.PI	<p>Value of π, approximately 3.142</p> <p>π is the ratio of a circle's circumference to the circle's diameter.</p>	Correct
Math.E	<p>Euler's number, approximately 2.718</p> <p>Euler's number is the base of natural logarithms.</p>	Correct
Math.LN2	<p>Natural logarithm of 2, approximately 0.693</p> <p>Math.LN10 is the natural log of 10.</p>	Correct
Math.LOG10E	<p>Base 10 logarithm of E, approximately 0.434</p> <p>Math.LOG2E is the base 2 log of E.</p>	Correct
Math.SQRT2	<p>Square root of 2, approximately 1.414</p> <p>Math.SQRT1_2 is the square root of 1/2.</p>	Correct

Reset

Math methods

The `Math` object has a range of trigonometric methods, including `sin()`, `cos()`, and `tan()`, and general calculation methods, including `log()` and `pow()`. Some commonly used `Math` methods are summarized in the table below.

Table 6.13.1: Common Math object methods.

Method	Description	Example
abs(x)	Returns the absolute value of x	<code>Math.abs(-5); // 5</code>
ceil(x)	Returns x rounded up to the nearest integer	<code>Math.ceil(2.1); // 3</code>
cos(x)	Returns the cosine of the radians x	<code>Math.cos(Math.PI) // -1</code>
floor(x)	Returns x rounded down to the nearest integer	<code>Math.floor(2.9) // 2</code>
log(x)	Returns the natural logarithm of x	<code>Math.log(Math.E) // 1</code>
max(n1, n2, n3, ...)	Returns the largest number	<code>Math.max(5, 2, 8, 1) // 8</code>
min(n1, n2, n3, ...)	Returns the smallest number	<code>Math.min(5, 2, 8, 1) // 1</code>
pow(x, y)	Returns x to the power of y	<code>Math.pow(2, 3) // 8</code>
round(x)	Returns x rounded to the nearest integer	<code>Math.round(3.5) // 4</code>
sin(x)	Returns the sine of radians x	<code>Math.sin(Math.PI) // 0</code>
sqrt(x)	Returns the square root of x	<code>Math.sqrt(25) // 5</code>
tan(x)	Returns the tangent of radians x	<code>Math.tan(Math.PI / 4) // 1</code>

[Feedback?](#)**PARTICIPATION
ACTIVITY**

6.13.2: Math methods.

Enter the value assigned to **x** in each code segment.

1) `x = Math.sin(Math.PI / 2);`

Correct

[Check](#)[Show answer](#)The sine of $\pi/2$ radians is 1.2) `x = Math.pow(2, 3);`[Check](#)[Show answer](#)**Correct**2 to the power of 3 = $2 * 2 * 2 = 8$.3) `x = Math.sqrt(9);`[Check](#)[Show answer](#)**Correct**

Square root of 9 is 3.

4) `x = Math.round(12.6) +
Math.floor(3.7);`[Check](#)[Show answer](#)**Correct**12.6 rounds to 13, and the floor of 3.7 is 3, so $13 + 3 = 16$.5) `x = Math.max(9, 14,
-26);`[Check](#)[Show answer](#)**Correct**

14 is larger than 9 and -26.

[Feedback?](#)

Producing random numbers

Many applications, especially games and simulations, need random numbers to simulate random processes. The **`Math.random()`** method returns a pseudo-random number ≥ 0 and < 1 . A **pseudo-random number** is a number generated by an algorithm that approximates randomness, but is not truly random.

Figure 6.13.1: Display 5 random numbers with `Math.random()`.

```
for (let i = 0; i < 5; i++) {
  console.log(Math.random());
}
```

```
0.5216294566239728
0.5399290004983317
0.05689844662407162
0.8711941395310085
0.7131957592778093
```

[Feedback?](#)

The figure below shows a `getRandomNumber()` function that performs the necessary calculations to generate a random integer between two integers.

Figure 6.13.2: Display five random numbers between 1 and 10.

```
// Return a random integer between min and max (inclusive).
function getRandomNumber(min, max) {
  return Math.floor(Math.random() * (max - min + 1)) +
min;
}

for (let i = 0; i < 5; i++) {
  console.log(getRandomNumber(1, 10));
}
```

```
7
3
1
8
4
```

[Feedback?](#)

PARTICIPATION ACTIVITY

6.13.3: Random numbers.



- 1) Numbers produced by `Math.random()` appear to be random.

- ☒ True
☐ False

Correct

`Math.random()` produces pseudo-random numbers that are not truly random but are suitable for many applications.



- 2) `Math.floor(Math.random() * 10)` produces a random

Correct

10 is not included. `Math.random()` returns a number less than 1. If `Math.random()`



number between 0 and 10, inclusive.

- ☐ True
- ☒ False

3) `Math.floor(Math.random() * 10) + 2` produces a random number between 2 and 11, inclusive.

- ☒ True
- ☐ False

generated 0.9999999 then `0.9999999 * 10 = 9.999999`, and `Math.floor(9.999999) = 9`.

Correct

`Math.floor(Math.random() * 10)` generates a random number between 0 and 9, inclusive. Adding 2 produces a random number between 2 and 11.



[Feedback?](#)

PARTICIPATION ACTIVITY

6.13.4: Practice with random numbers.



The `displayCard(rank, suit)` function displays a playing card to the console, given the rank (1-13) and suit (0-3).

Write a for loop that calls `displayCard()` 10 times, each time with a random rank and suit.

```
1 // rank should be a number between 1 and 13, and suit between
2 function displayCard(rank, suit) {
3     switch (rank) {
4         case 1: rank = "A"; break;
5         case 10: rank = "T"; break;
6         case 11: rank = "J"; break;
7         case 12: rank = "Q"; break;
8         case 13: rank = "K"; break;
9         default: if (rank < 1 || rank > 13) {
10             console.log("Bad rank value: " + rank);
11         }
12     }
13
14     switch (suit) {
15         case 0: suit = "♥"; break;
16         case 1: suit = "♦"; break;
```

Run JavaScript

[Reset code](#)

Your console output

5♥

5♥

K♦

K♦

A♣

A♣

T♠

T♠

▼ View solution

 Explain

--- START FILE: JavaScript ---

```
function displayCard(rank, suit) {
  switch (rank) {
    case 1: rank = "A"; break;
    case 10: rank = "T"; break;
    case 11: rank = "J"; break;
    case 12: rank = "Q"; break;
    case 13: rank = "K"; break;
    default: if (rank < 1 || rank > 13) {
      console.log("Bad rank value: " + rank);
    }
  }

  switch (suit) {
    case 0: suit = "♥"; break;
    case 1: suit = "♦"; break;
    case 2: suit = "♣"; break;
    case 3: suit = "♠"; break;
    default: console.log("Bad suit value: " + suit);
  }

  console.log("┌──────────┐");
  console.log("│" + rank + suit + "│");
  console.log("│          │");
  console.log("│          │" + rank + suit + "│");
```

```

    console.log("_____");
}

displayCard(5, 0);
displayCard(13, 1);
displayCard(1, 2);
displayCard(10, 3);

function getRandomNumber(min, max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
}

for (let i = 0; i < 10; i++) {
    let rank = getRandomNumber(1, 13);
    let suit = getRandomNumber(0, 3);
    displayCard(rank, suit);
}

--- END FILE: JavaScript ---

```

[Feedback?](#)**CHALLENGE
ACTIVITY**

6.13.1: Math object.



530096.4000608.qx3zqy7

Start

1

Compute: $z = \sqrt{x * y}$ 

2

```

1 // Your code will be tested with x = 4 and y = 3, and other values
2 let x = 4;
3 let y = 3;
4 let z = /* Your solution goes here */;
5
6 // Output z with two decimal places
7 console.log(z.toFixed(2));

```



3

1

2

3