Number

A number that can be positive, negative, or zero.

The `Number` data type is useful for describing values such as position, size, and color. A number can be an integer such as 20 or a decimal number such as 12.34. For example, a circle's position and size can be described by three numbers:

```
circle(50, 50, 20);
```

```
circle(50, 50, 12.34);
```

Numbers support basic arithmetic and follow the standard order of operations: Parentheses, Exponents, Multiplication, Division, Addition, and Subtraction (PEMDAS). For example, it's common to use arithmetic operators with p5.js' system variables that are numbers:

```
// Draw a circle at the center.
circle(width / 2, height / 2, 20);
```

```
// Draw a circle that moves from left to right.
circle(frameCount * 0.01, 50, 20);
```

Here's a quick overview of the arithmetic operators:

```
1 + 2 // Add
1 - 2 // Subtract
1 * 2 // Multiply
1 / 2 // Divide
1 % 2 // Remainder
1 ** 2 // Exponentiate
```

It's common to update a number variable using arithmetic. For example, an object's location can be updated like so:

```
x = x + 1;
```

The statement above adds 1 to a variable `x` using the `+` operator. The addition assignment operator `+=` expresses the same idea:

```
x += 1;
```

Here's a quick overview of the assignment operators:

```
x += 2 // Addition assignment
x -= 2 // Subtraction assignment
x *= 2 // Multiplication assignment
x /= 2 // Division assignment
x %= 2 // Remainder assignment
```

Numbers can be compared using the relational operators `>`, `<< code="">, >=, <=< code="">, ===, and !==. For example, a sketch's frameCount can be used as a timer:`

```
if (frameCount > 1000) {
  text('Game over!', 50, 50);
}
```

An expression such as `frameCount > 1000` evaluates to a `Boolean` value that's either `true` or `false`. The relational operators all produce `Boolean` values:

```
2 > 1 // true
2 < 1 // false
2 >= 2 // true
2 <= 2"" true="" !="=" false="" <="" code=""/>
```

See Boolean for more information about comparisons and conditions.

Note: There are also `==` and `!=` operators with one fewer `=`. Don't use them.

Expressions with numbers can also produce special values when something goes wrong:

```
sqrt(-1) // NaN
1 / 0 // Infinity
```

The value `NaN` stands for Not-A-Number. `NaN` appears when calculations or conversions don't work. `Infinity` is a value that's larger than any number. It appears during certain calculations.

Examples

```
function setup() {
  createCanvas(100, 100);

  background(200);

  // Draw a circle at the center.
  circle(50, 50, 70);

  // Draw a smaller circle at the center.
  circle(width / 2, height / 2, 30);

  describe('Two concentric, white circles drawn on a gray
background.');
}
```

```
function setup() {
  createCanvas(100, 100);

  describe('A white circle travels from left to right on a
gray background.');
}

function draw() {
  background(200);

  circle(frameCount * 0.05, 50, 20);
}
```

This page is generated from the comments in src/core/reference.js . Please feel free to edit it and submit a pull request!

Related References

class
A template for creating objects of a particular type.

console
Prints a message to the web browser's console.

for
A way to repeat a block of code when the number of iterations is known.

function
A named group of statements.