

Reference > mult()

mult()

Multiplies a vector's **x**, **y**, and **z** components.

`mult()` can use separate numbers, as in `v.mult(1, 2, 3)`, another `p5.Vector` object, as in `v.mult(v2)`, or an array of numbers, as in `v.mult([1, 2, 3])`.

If only one value is provided, as in `v.mult(2)`, then all the components will be multiplied by 2. If a value isn't provided for a component, it won't change. For example, `v.mult(4, 5)` multiplies `v.x` by 4, `v.y` by 5, and `v.z` by 1. Calling `mult()` with no arguments, as in `v.mult()`, has no effect.

The static version of `mult()`, as in `p5.Vector.mult(v, 2)`, returns a new `p5.Vector` object and doesn't change the originals.

Examples

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

  background(200);

  // Style the points.
  strokeWeight(5);

  // Top-left.
  let p = createVector(25, 25);
  point(p);

  // Center.
  // Multiply all components by 2.
  p.mult(2);
  point(p);

  describe('Two black dots drawn on a gray square. One dot is in the top left corner and the other is in the center.');
```

```
function setup() {
  strokeWeight(5);

  // Top-left.
  let p = createVector(25, 25);
  point(p);

  // Bottom-right.
  // Multiply p.x * 2 and p.y * 3
  p.mult(2, 3);
  point(p);

  describe('Two black dots drawn on a gray square. One dot is in the top left corner and the other is in the bottom center.');
```

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

  background(200);

  // Style the points.
  strokeWeight(5);

  // Top-left.
  let p = createVector(25, 25);
  point(p);

  // Bottom-right.
  // Multiply p.x * 2 and p.y * 3
  let arr = [2, 3];
  p.mult(arr);
  point(p);

  describe('Two black dots drawn on a gray square. One dot is in the top left corner and the other is in the bottom center.');
```

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

  background(200);

  // Style the points.
  strokeWeight(5);

  // Top-left.
  let p = createVector(25, 25);
  point(p);

  // Bottom-right.
  // Multiply p.x * p2.x and p.y * p2.y
  let p2 = createVector(2, 3);
  p.mult(p2);
  point(p);

  describe('Two black dots drawn on a gray square. One dot is in the top left corner and the other is in the bottom center.');
```

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

  background(200);

  // Style the points.
  strokeWeight(5);

  // Top-left.
  let p = createVector(25, 25);
  point(p);

  // Bottom-right.
  // Create a new p5.Vector with
  // p3.x = p.x * p2.x
  // p3.y = p.y * p2.y
  let p2 = createVector(2, 3);
  let p3 = p5.Vector.mult(p, p2);
  point(p3);

  describe('Two black dots drawn on a gray square. One dot is in the top left corner and the other is in the bottom center.');
```

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

  describe('Two arrows extending from the top left corner. The blue arrow is twice the length of the red arrow.');
```

```
function draw() {
  background(200);

  let origin = createVector(0, 0);

  // Draw the red arrow.
  let v1 = createVector(25, 25);
  drawArrow(origin, v1, 'red');

  // Draw the blue arrow.
  let v2 = p5.Vector.mult(v1, 2);
  drawArrow(origin, v2, 'blue');
}
```

```
// Draws an arrow between two vectors.
function drawArrow(base, vec, myColor) {
  push();
  stroke(myColor);
  strokeWeight(3);
  fill(myColor);
  translate(base.x, base.y);
  line(0, 0, vec.x, vec.y);
  rotate(vec.heading());
  let arrowSize = 7;
}
```

Syntax

- `mult(n)`
- `mult(x, y, [z])`
- `mult(arr)`
- `mult(v)`
- `mult(x, y, [z])`
- `mult(v, n, [target])`
- `mult(v0, v1, [target])`
- `mult(v0, arr, [target])`

Parameters

n	Number: The number to multiply with the vector
x	Number: Number to multiply with the x component of the vector.
y	Number: number to multiply with the y component of the vector.
z	Number: number to multiply with the z component of the vector.
arr	Number[]: array to multiply with the components of the vector.
v	p5.Vector: vector to multiply with the components of the original vector.
target	p5.Vector: vector to receive the result.
v0	p5.Vector:
v1	p5.Vector:

This page is generated from the comments in [src/math/p5.Vector.js](#). Please feel free to edit it and submit a pull request!

Related References

add Adds to a vector's x, y, and z components.	angleBetween Calculates the angle between two vectors.	array Returns the vector's components as an array of numbers.	clampToZero Replaces the components of a p5.Vector that are very close to zero with zero.
--	--	---	---