V= createVector(3,4).

Console

① Constructor function

② p1
↳ x
↳ y
↳ prototype

p2
↳ x
↳ y
↳ Prototype

particle.prototype
↳ show()
↳ __proto__

prototype chaining

Implemented

down

```
function Particle(){
    this.x = 100;
    this.y = 99;
}
var p = new particle();
  object
  ↳ x
  ↳ y
  ↳ __proto__: object
```

```
function particle(){
  //
  this.show = function(){
  point(this.x, this.y);
  }
  var p;
  function setup(){
   createCanvas(600,300);
   p = new Particle();
  }
```

```
p
↳ show: ()
x: 100
y: 99
↳ __proto__: object
```

```
var p2 = new particle()
   p2.x++
p2
 ↳ show()
   x: 101
   y: 99
 ↳ __proto__: object
```

```
~~function(){~~ function particle(){
     //          // not show function.
  }
particle.prototype.show = ~~show~~ function(){
      point(this.x, this.y);
  }
var p;
// rest code remain same.
```

```
p
↳ x
↳ y
↳ __proto__: object
   ↳ constructor
       particle
   ↳ show: ()
   ↳ __proto__: object.
```

now see ②

```
V = createVector(3,4);
```

⇒ Interesting

hasOwnProperty

```
p.hasOwnProperty('z');
p.hasOwnProperty('show');
```

false

↳ properties individual to that particular instance

this is prototype chain

Console

```
→ V.mag()
→ V.mult(2)
→ V.double();
```

Uncaught type error

```
↳ p5.p5
↳ x:6
↳ y:8
↳ z:0
```

all function are in

proto : object

all these proprties are for that particular instane

```
p5.Vector.prototype.doubl
    =function(){
  this.x *= 2;
  this.y *= 2;
  this.z *= 2;
  }
```