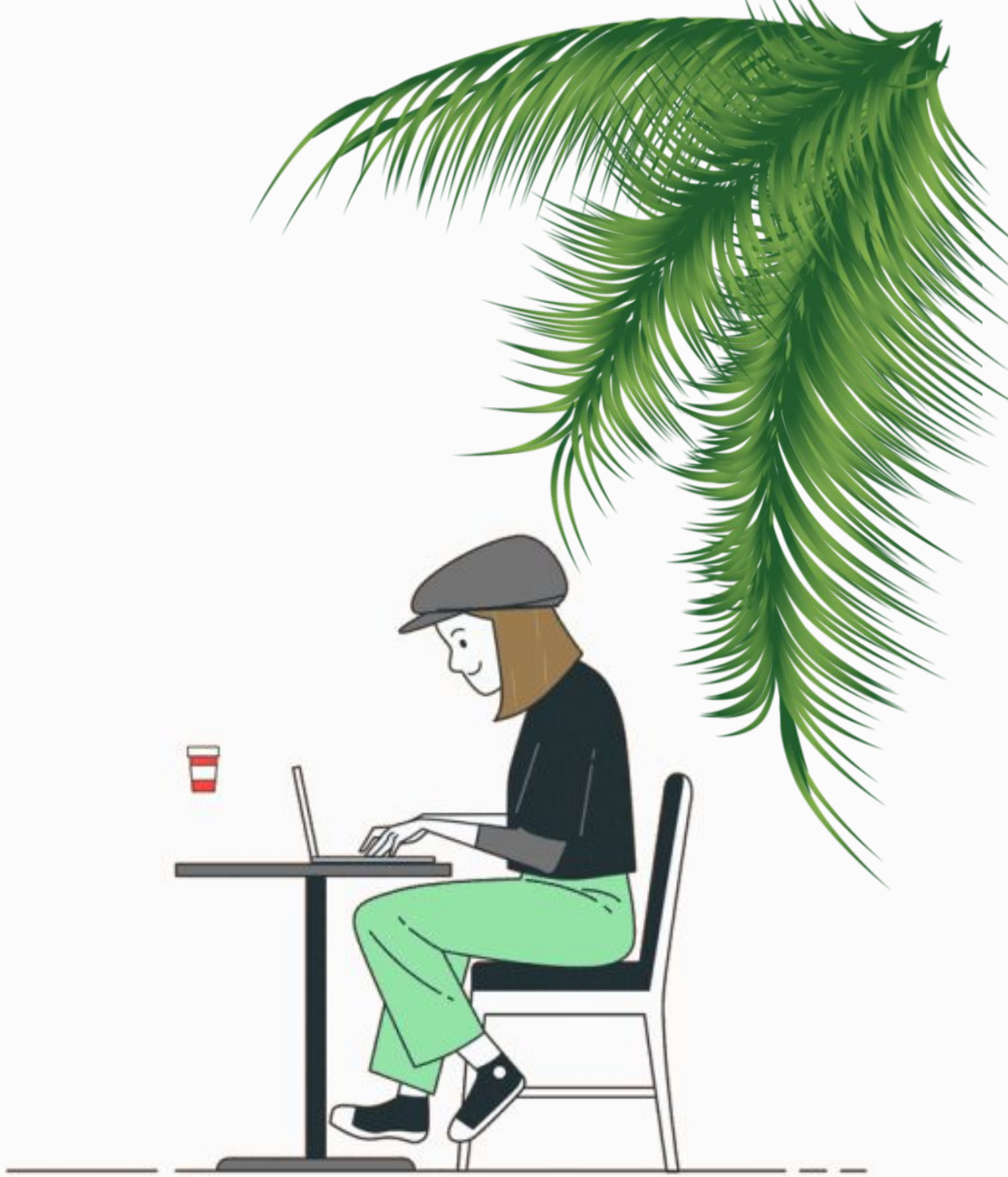
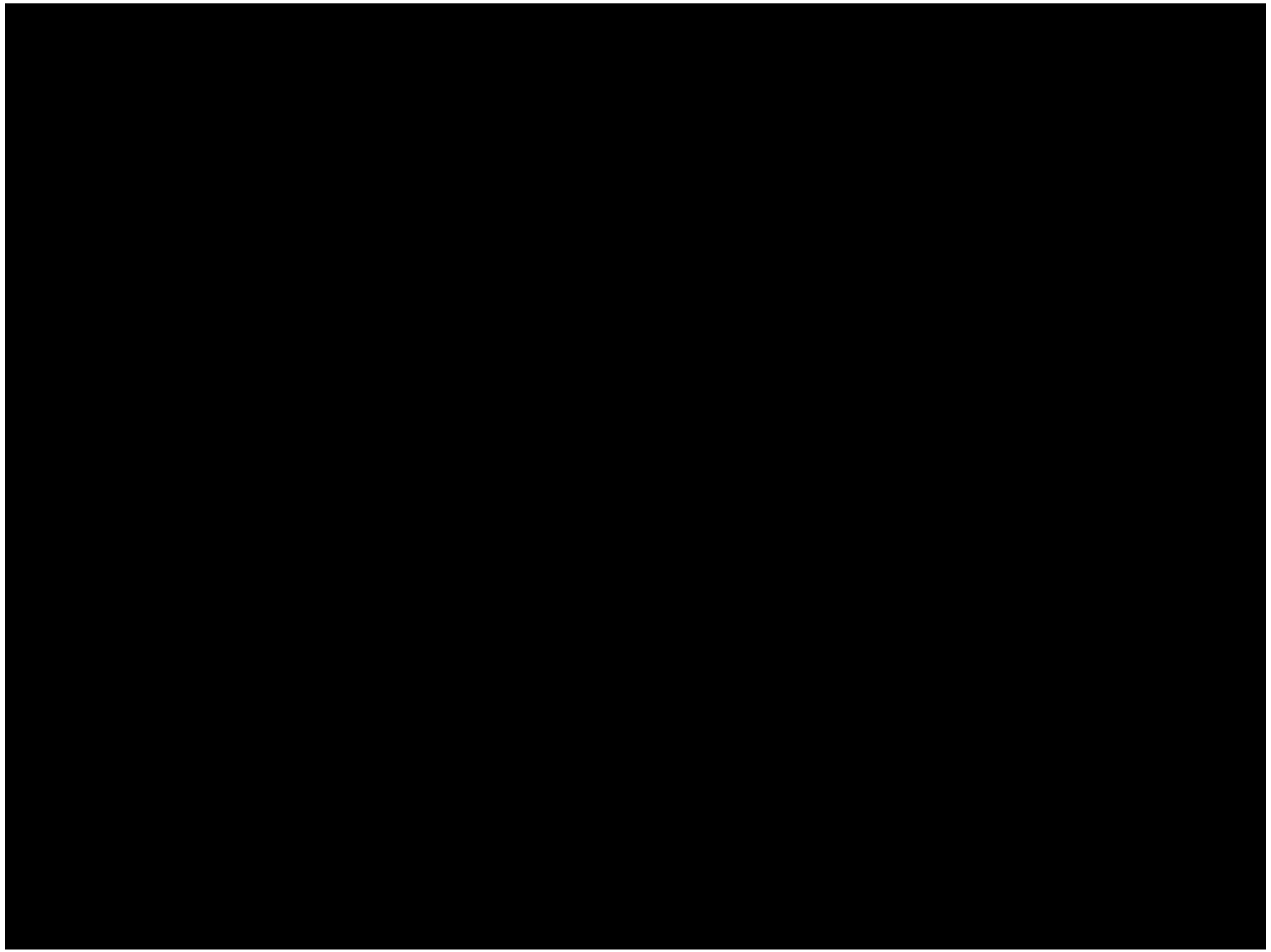
map()
forEach()
filter()
find()
reduce()
Class
Object
Inheritance
Prototypical inheritance



array.map()

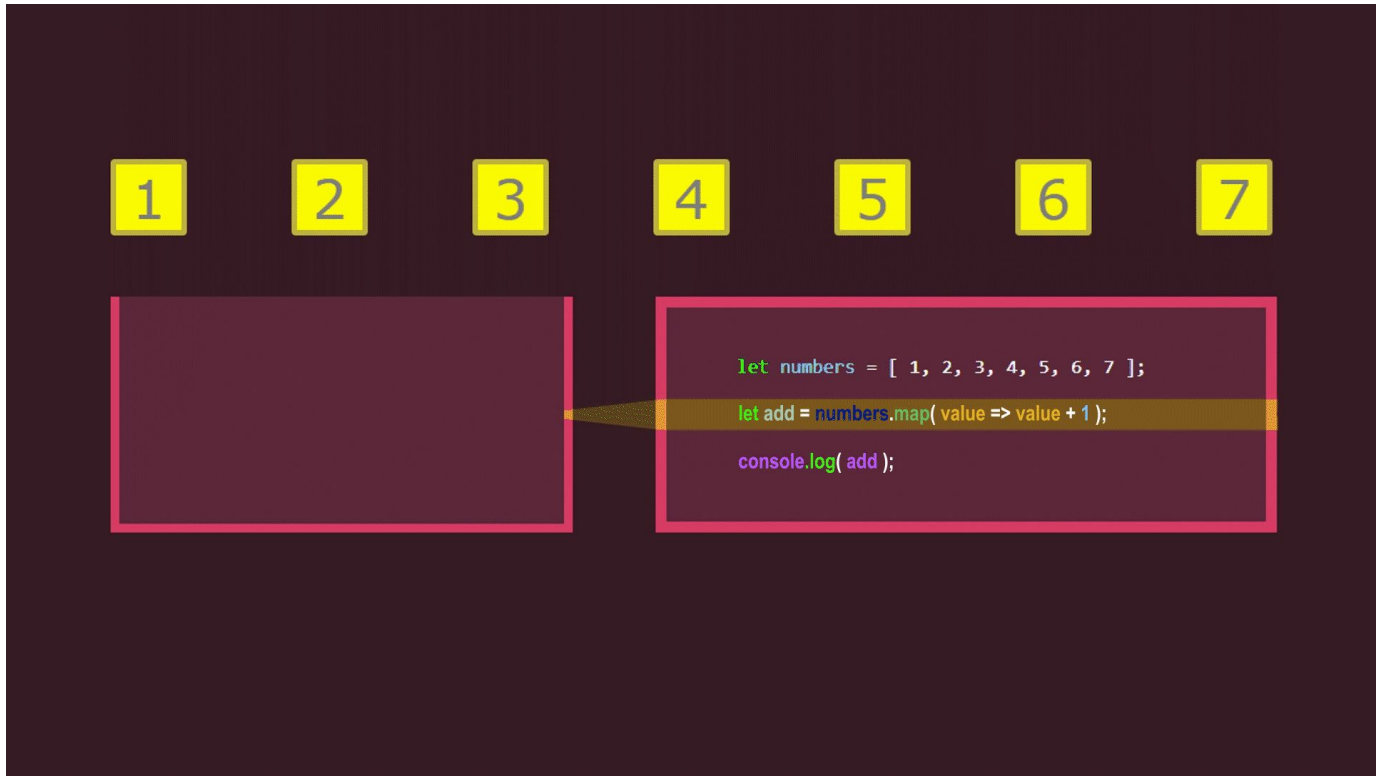


How Array map method works?

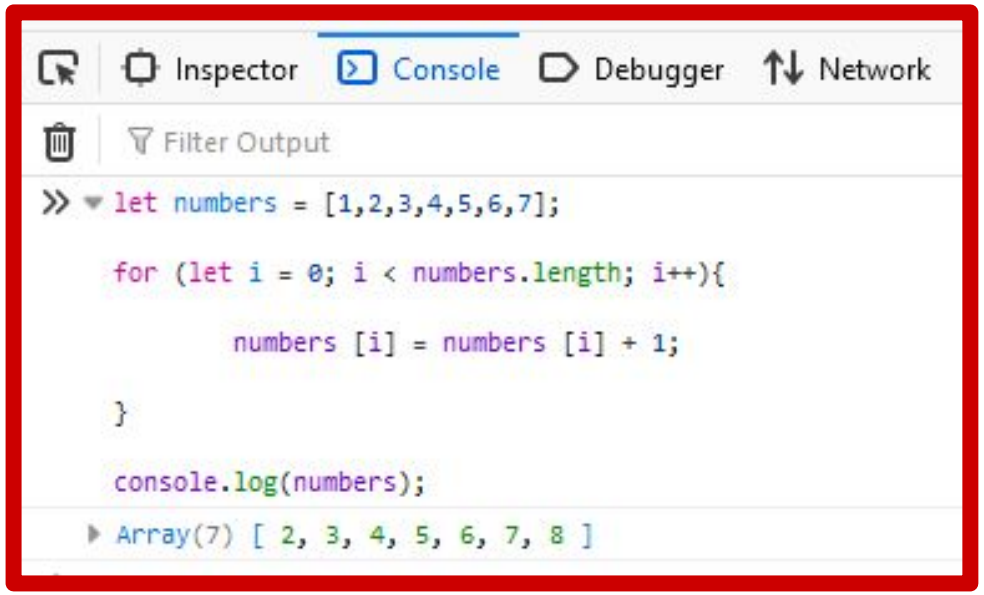
INPUT [  ]

Ref: tyc-wordpress.s3.amazonaws.com

Array map to do one line loop magic



Array loop



```
1   var arr=[1,2,,4,5];
2
3   arr.forEach(e=>console.log(e))
4
```

PROBLEMS TERMINAL ...

```
1
2
4
5
```

forEach	map
Return value: **undefined**	Return value: **newArray will be created** based on your callback function
Original Array: **not modified**	Original Array: **not modified**


for
forEach
map
?
while
for of

6 Ways to loop through an array

1. for loop

```
let array = [10,20,30,40];

for (let i = 0; i <= array.length; i++)
{
    console.log(array[i]);
}
```

2. do...while

```
let i = 0;
do {
  console.log(i);
  i++;
} while (i < 5)
```

3. while loop

```
let array = [10,20,30,40];
let i = 0;
while (i < array.length){
    console.log(array[i]);
    i++;
};
```

4. for...of

```
let array = [10,20,30,40];

for (let i of array)
{
    console.log(i);
}
```

5. map() method

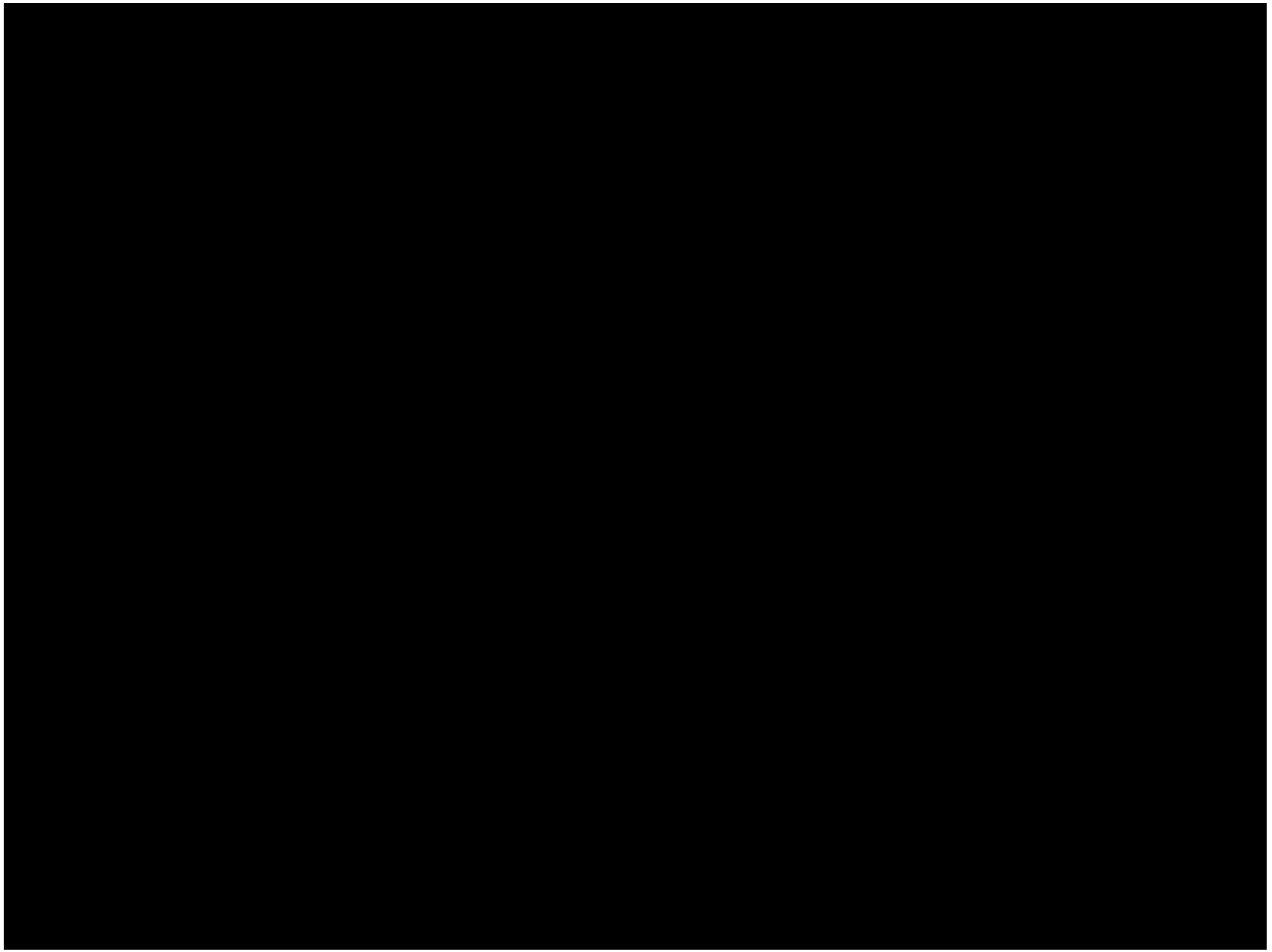
```
const names = ['Lincoln', 'Daniel', 'Tabitha', 'McLeod']

let newArr = names.map((index) => console.log(index));
```

6. forEach()

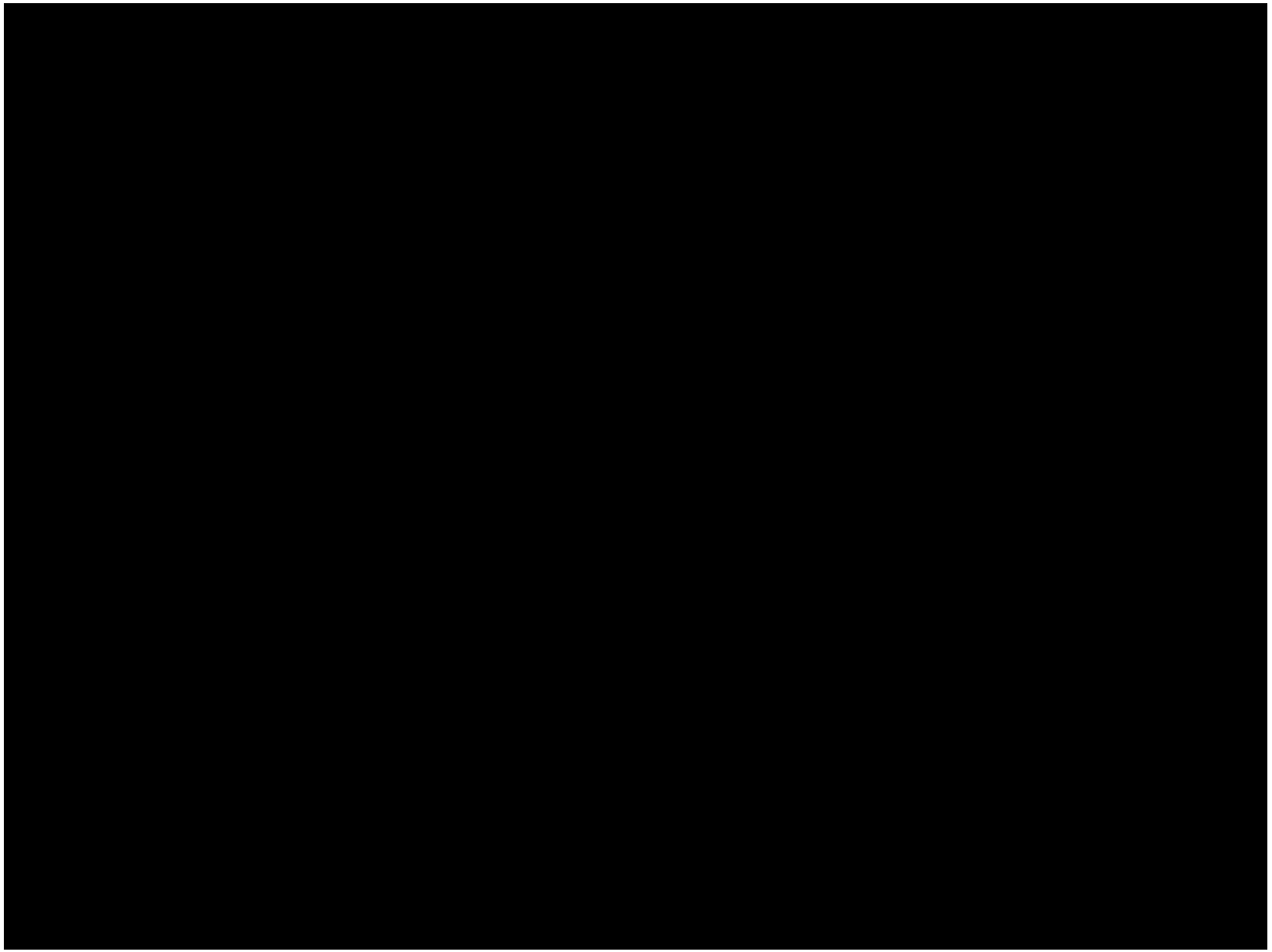
```
let array = [10,20,30,40];

array.forEach((index) => {console.log(index)});
```

What is filtering?

array.filter()

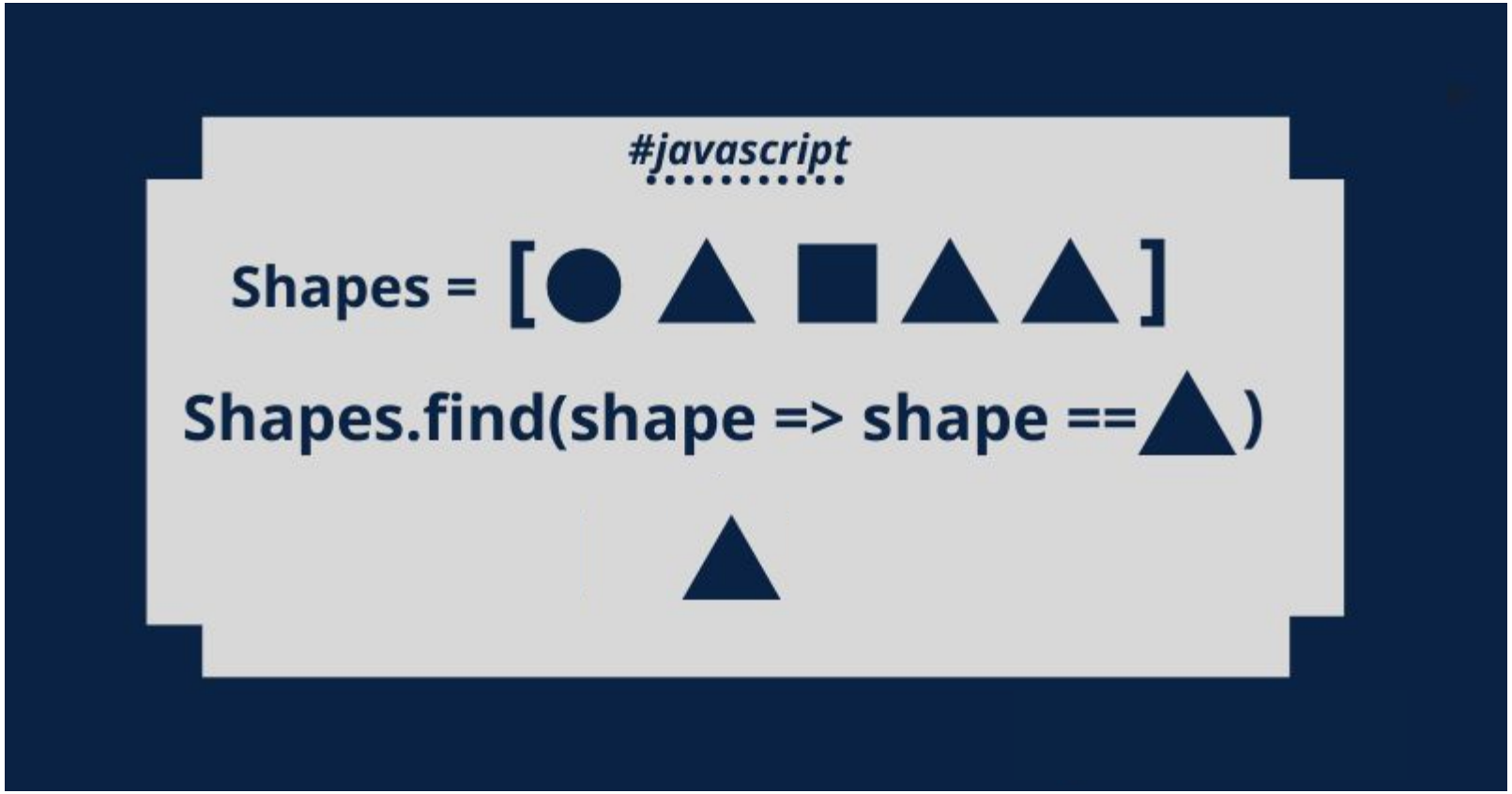


array.filter()

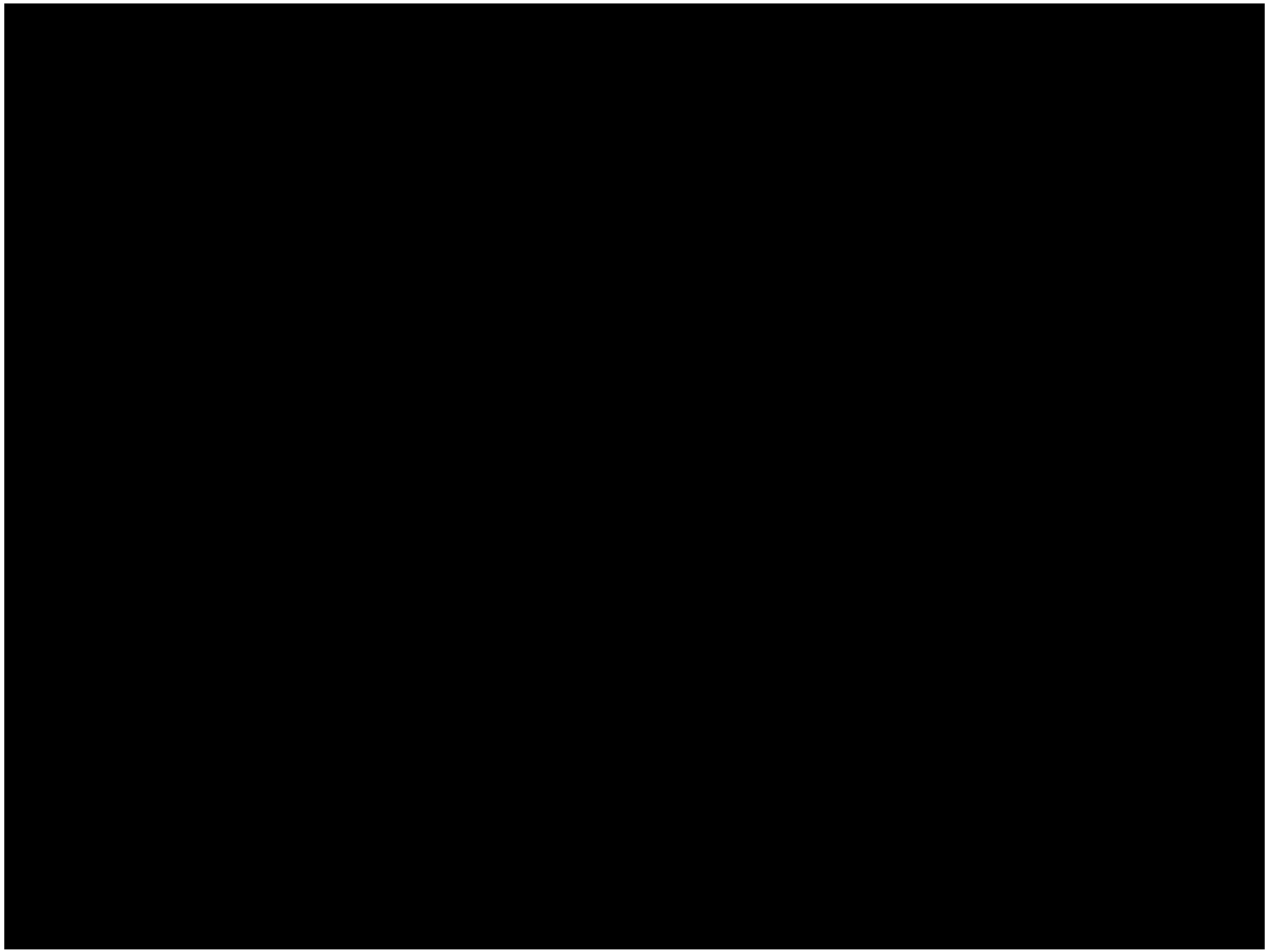


What are you finding?

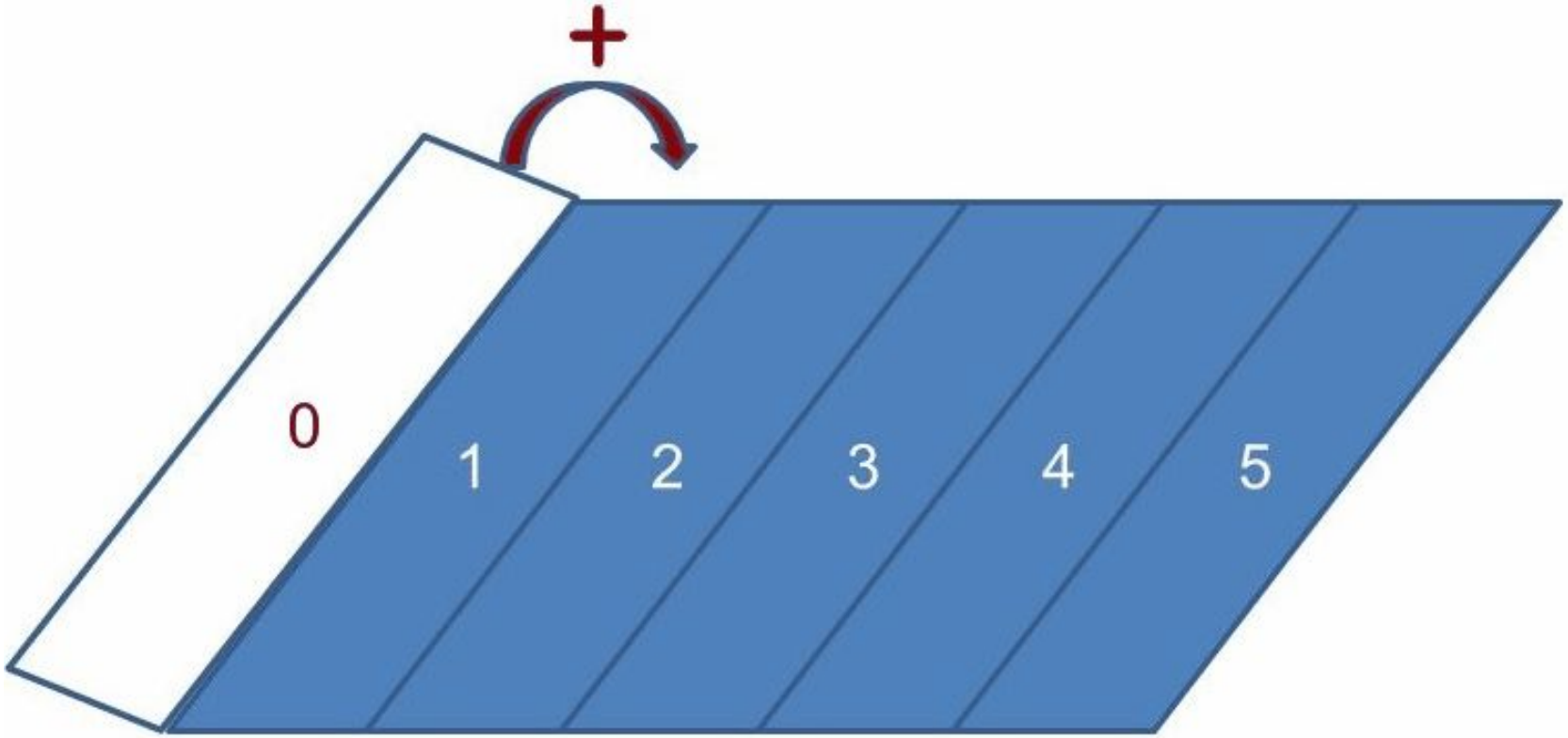
array.find()



array.find()



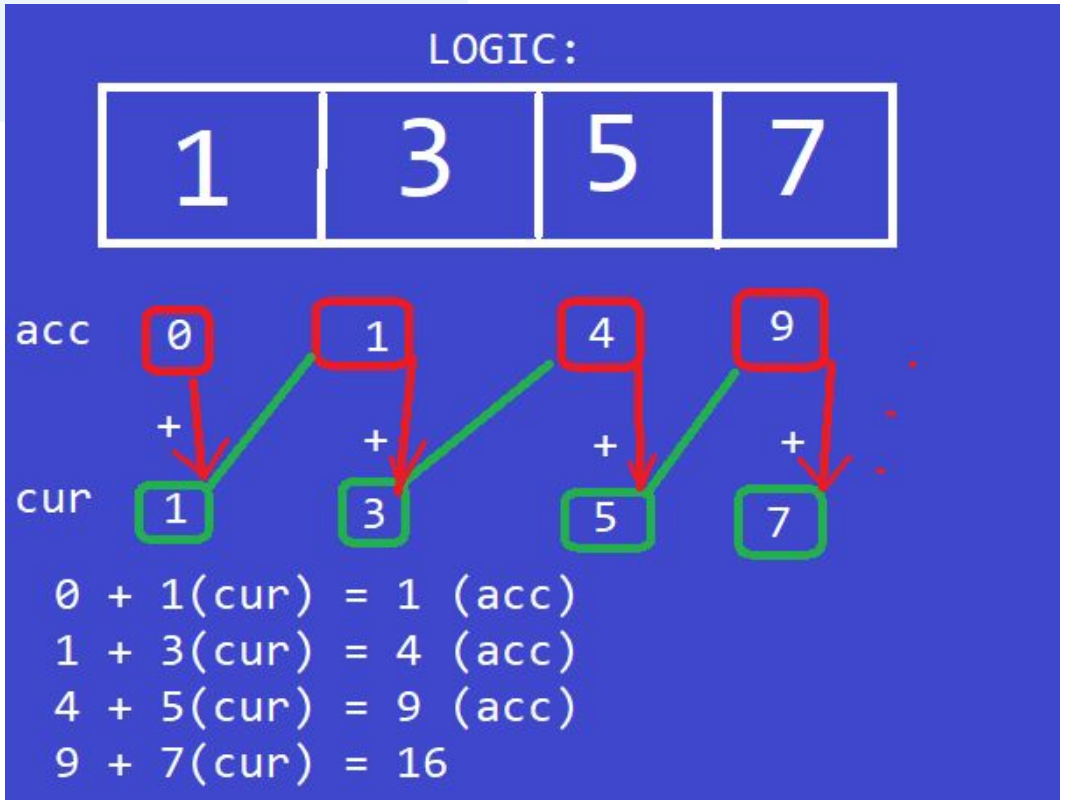
array.reduce()



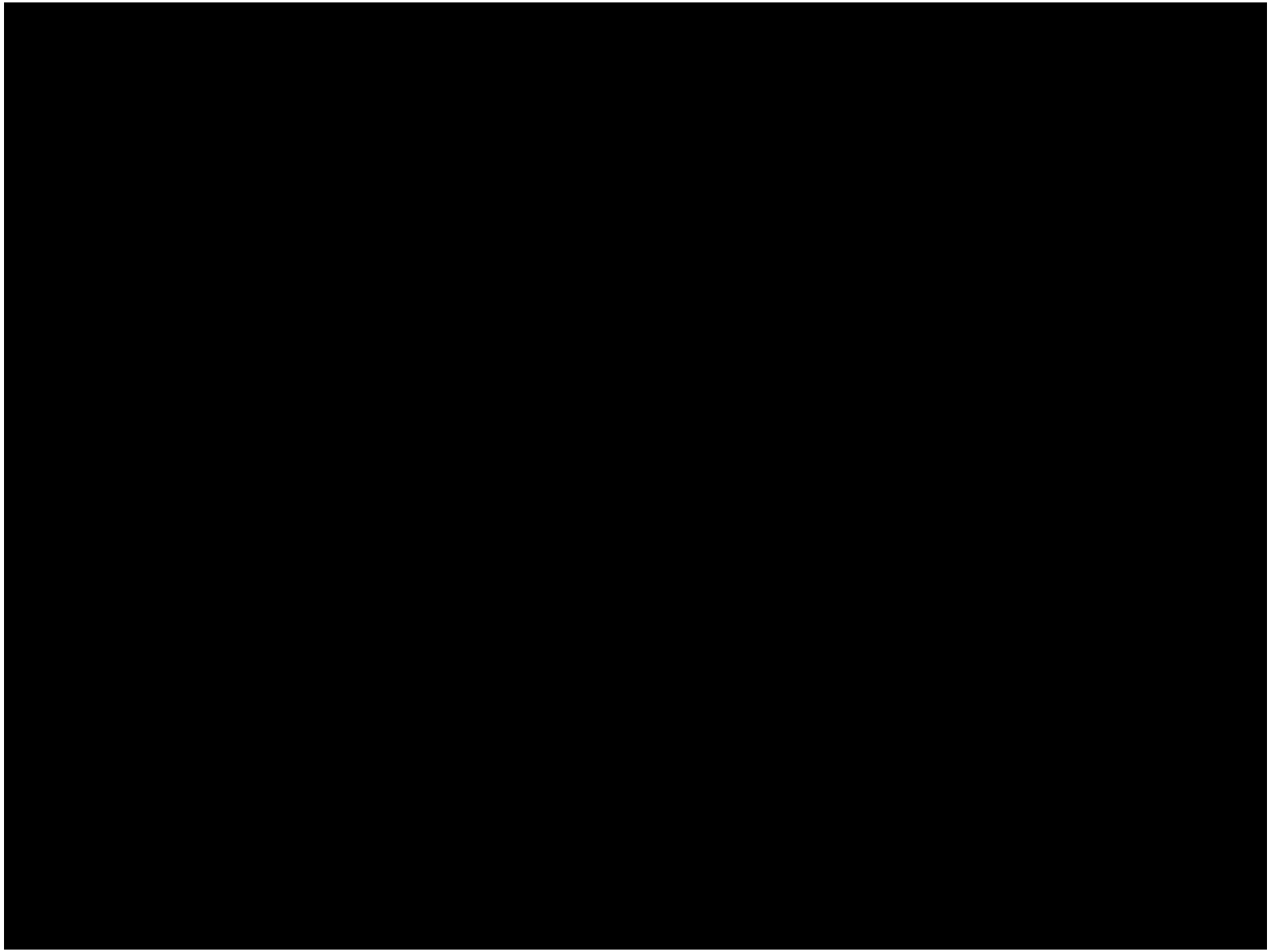
Ref: freecodecamp

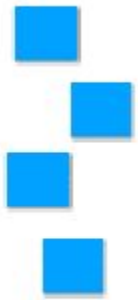
array.reduce()

```
const newNumbers = [1, 3, 5, 7];

const newSum = newNumbers.reduce((accumulator, currentValue) => {

  return (accumulator + currentValue);

}, 0 );
```

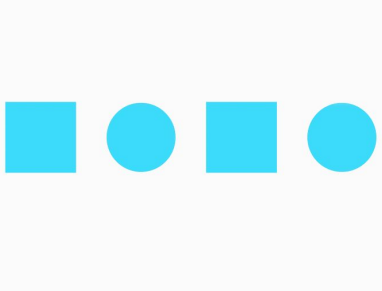


array.reduce()





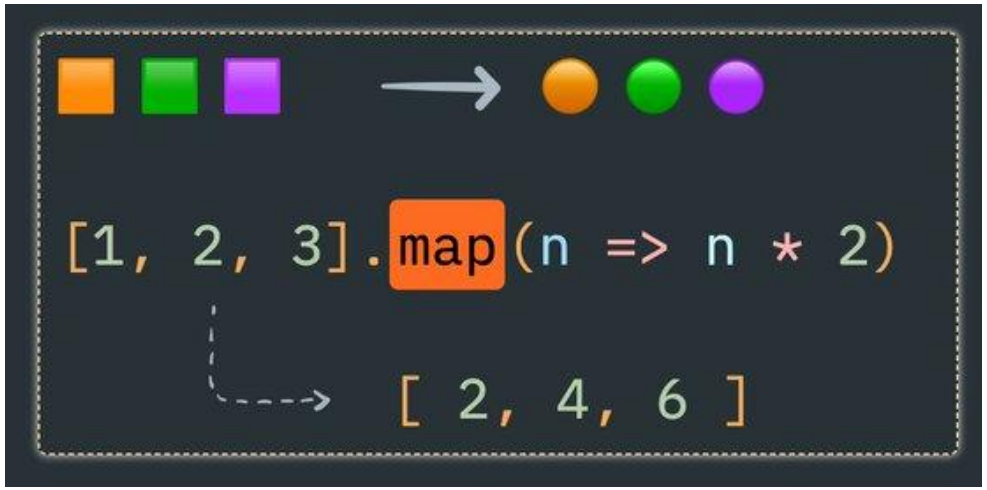
Map

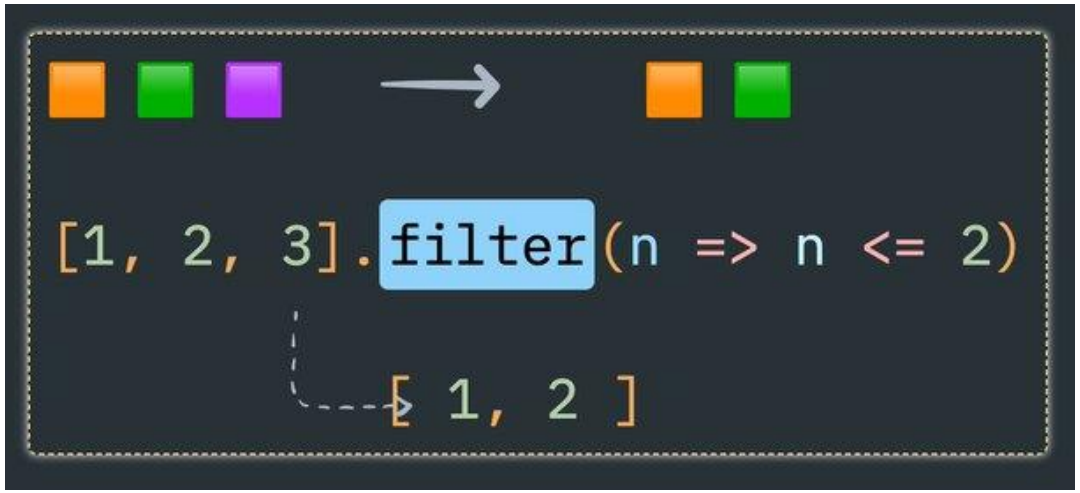


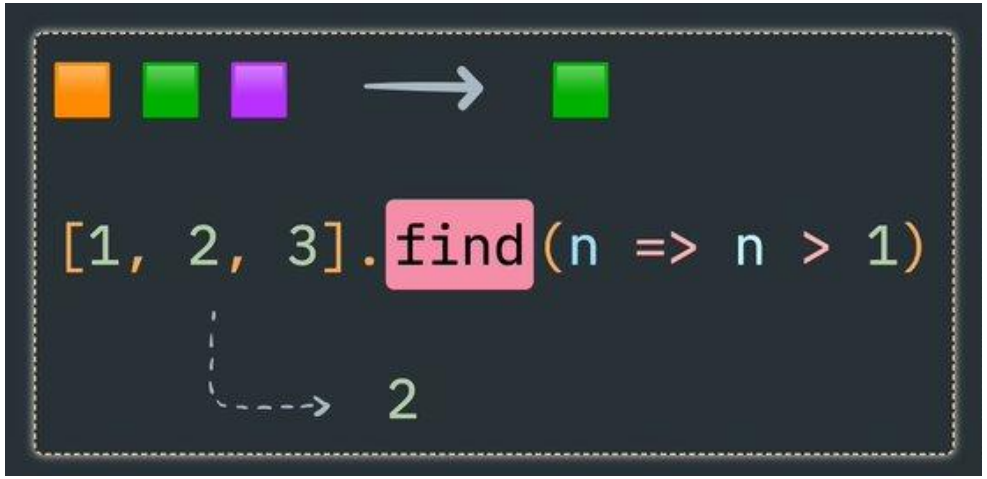
Filter

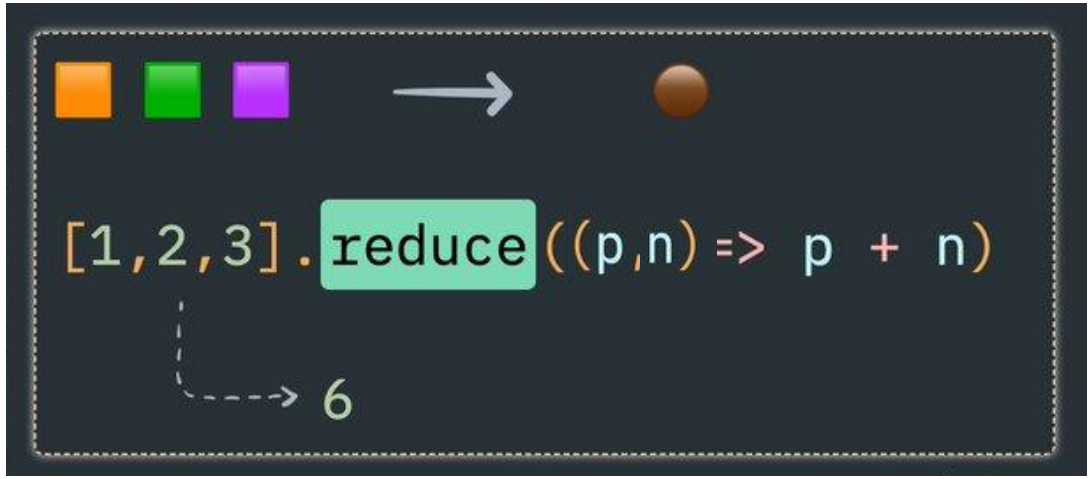
Find

Reduce


[1, 2, 3].map(n => n * 2)
[2, 4, 6]


[1, 2, 3].filter(n => n <= 2)
[1, 2]


[1, 2, 3].find(n => n > 1)
2


[1,2,3].reduce((p,n) => p + n)
6

Bicycle Class

Bicycle 1
Object

Properties

Color (Green)

Name (Hiland)

Methods

start ()
break ()
stop ()

Bicycle 2
Object

Properties

Color (Red)

Name (EuroBike)

Methods

start ()
break ()
stop ()

Bicycle 3
Object

Properties

Color (Blue)

Name (TrekBike)

Methods

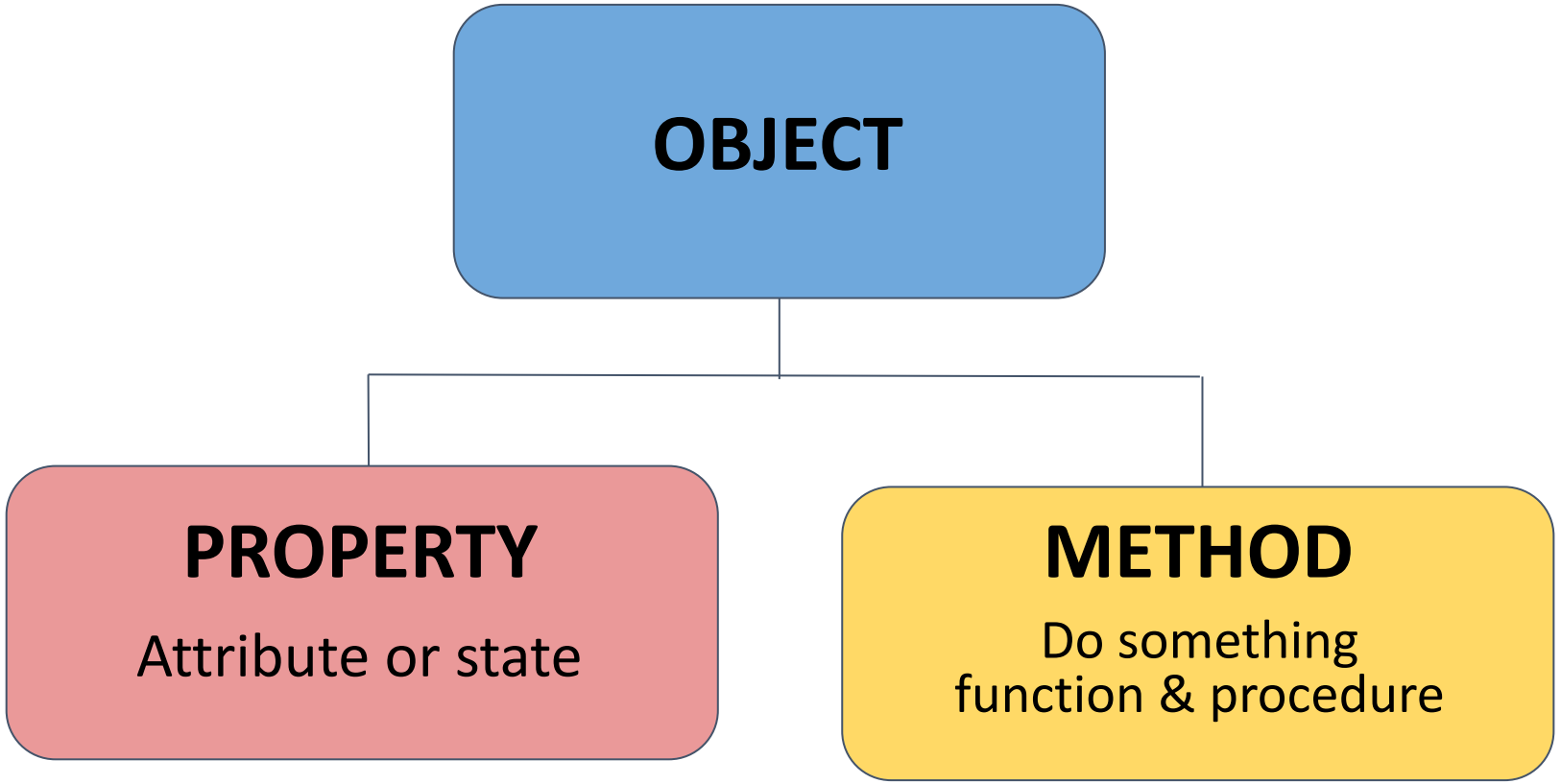
start ()
break ()
stop ()

What is an Object?

Class

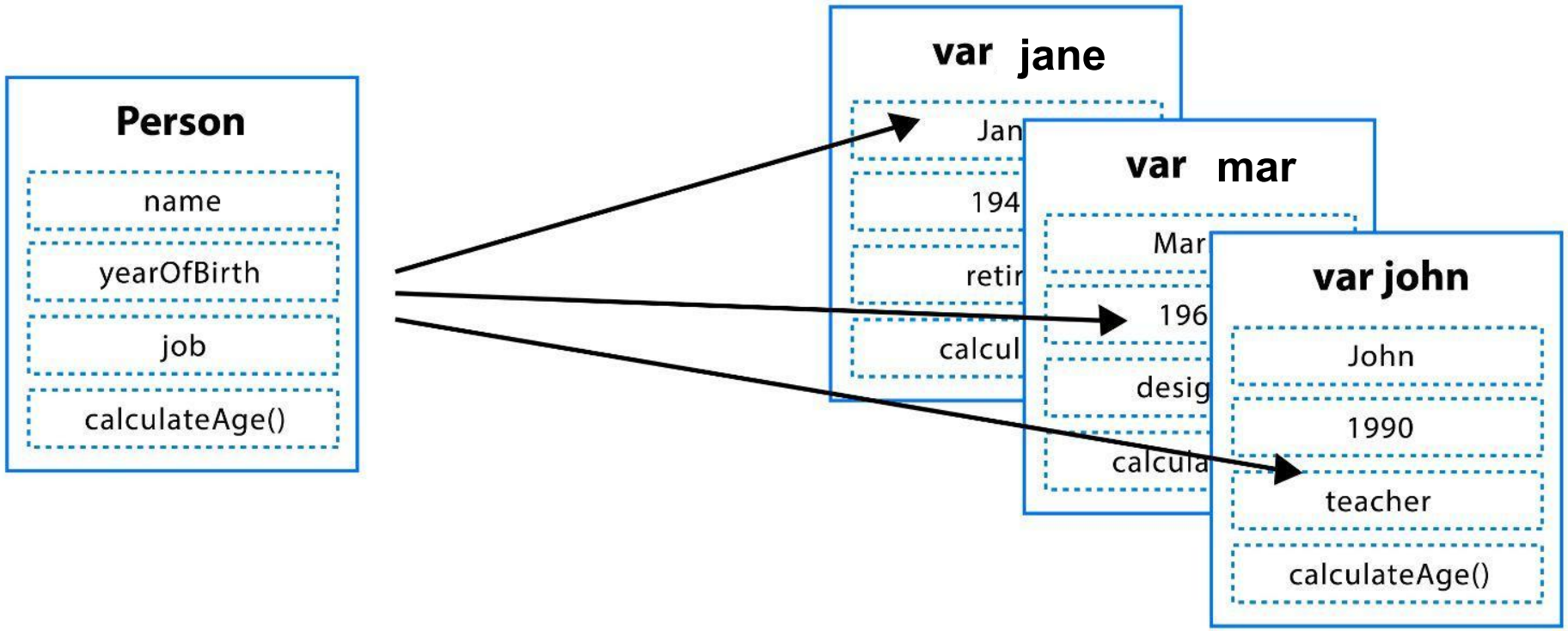
Template for making Objects

Bike Blueprint

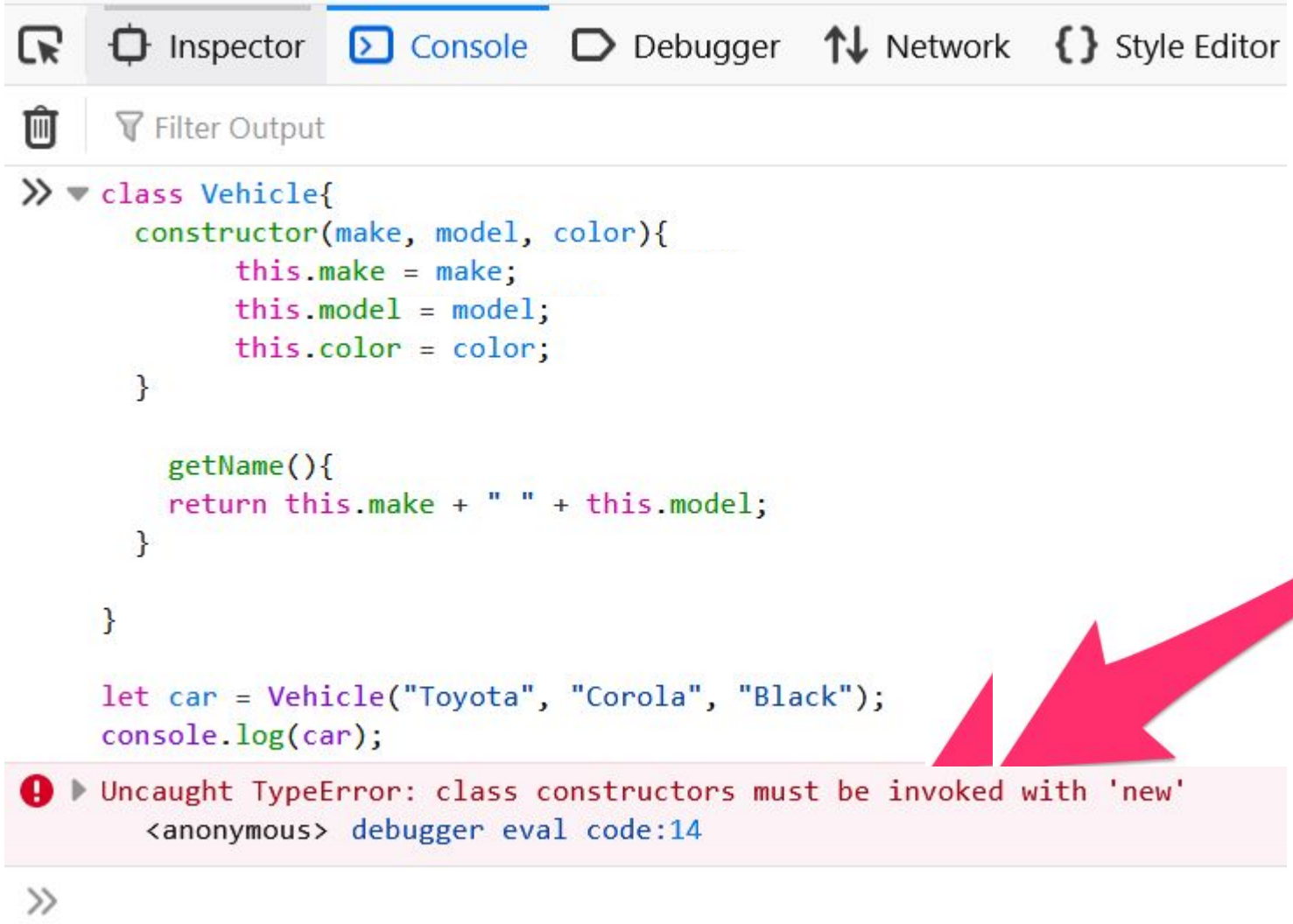

OBJECT
PROPERTY
Attribute or state
METHOD
Do something
function & procedure

CONSTRUCTOR

INSTANCES



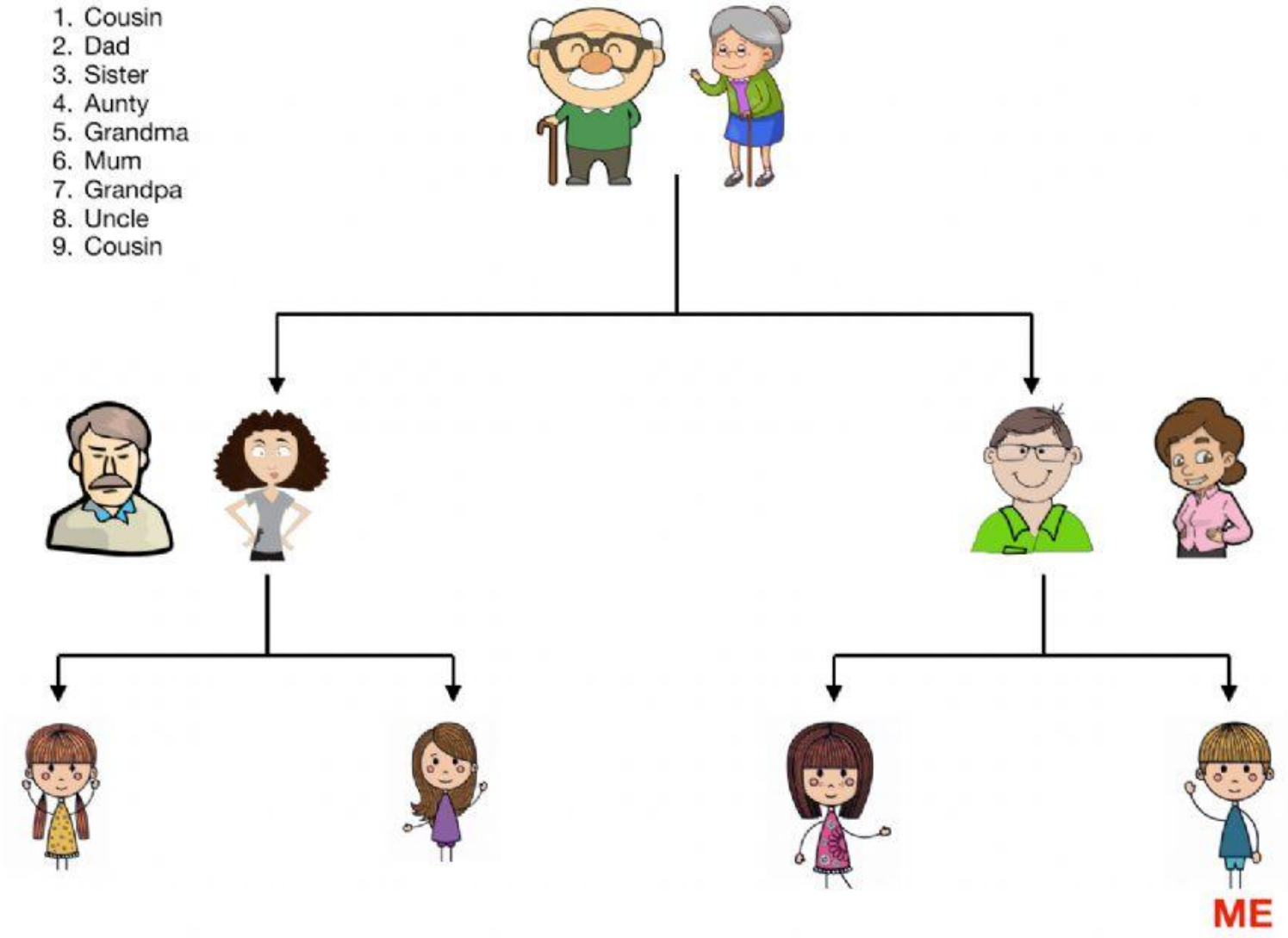
Creating object with constructor



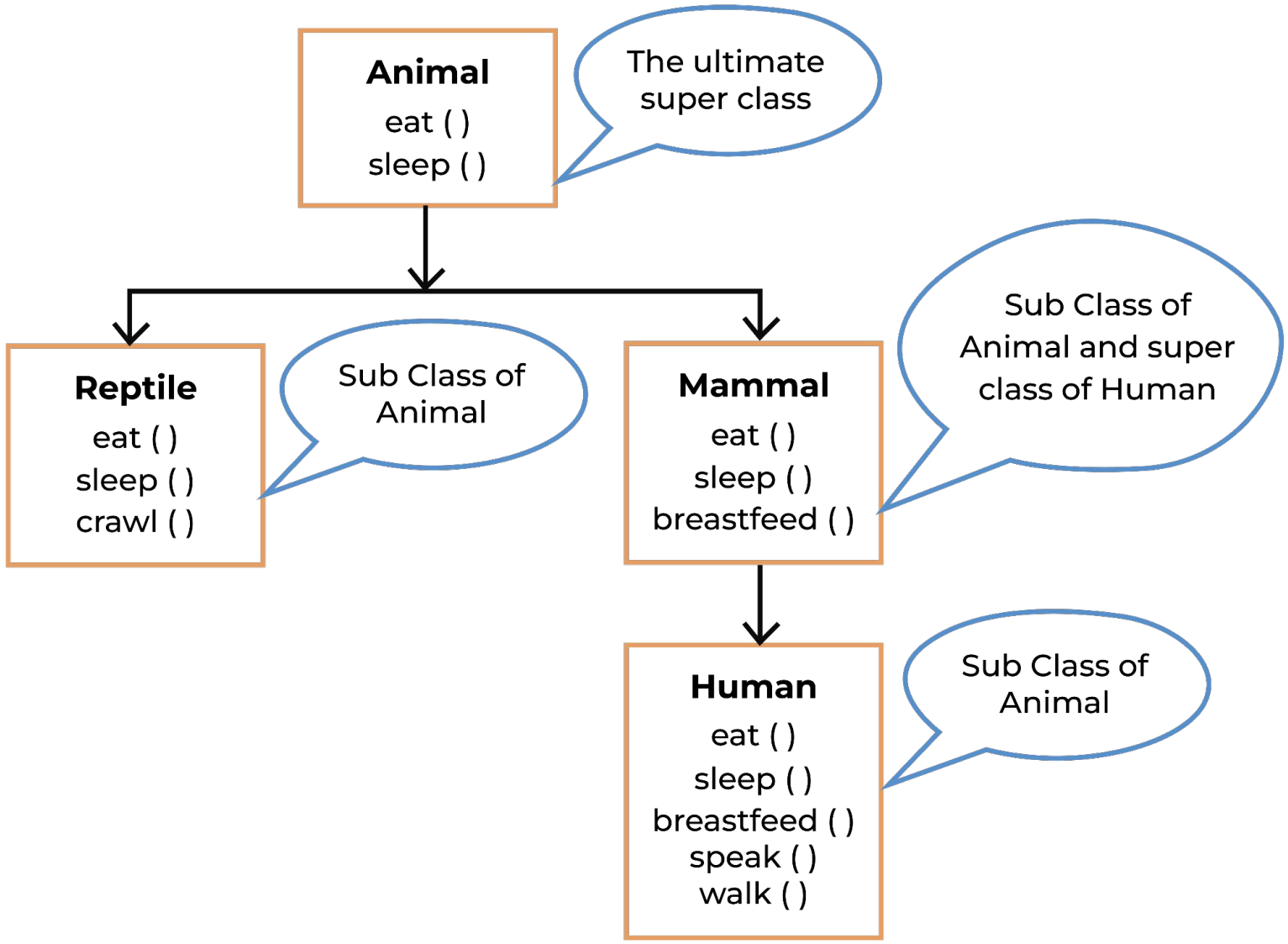
Creating object with constructor

Family tree

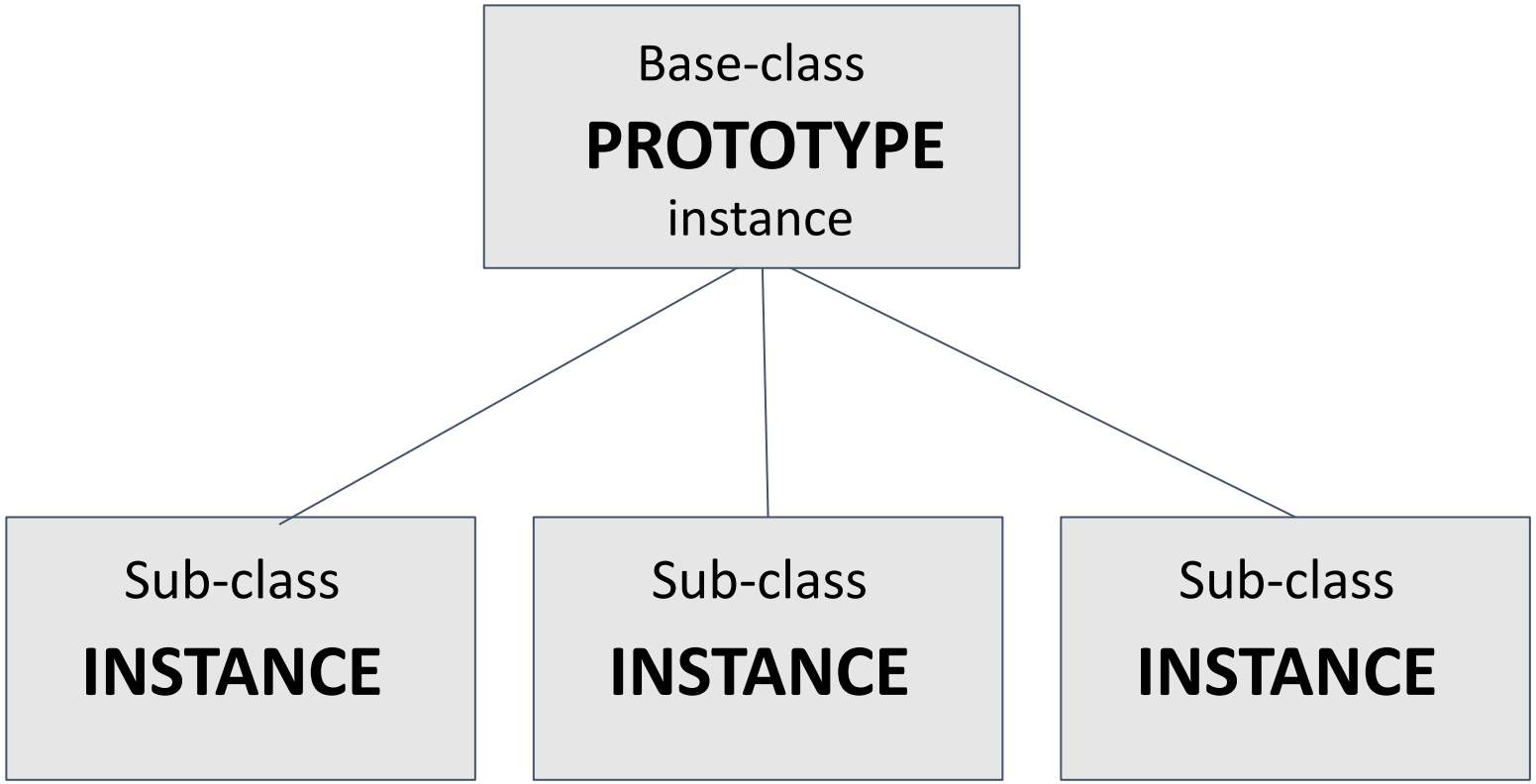
1. Cousin
2. Dad
3. Sister
4. Aunty
5. Grandma
6. Mum
7. Grandpa
8. Uncle
9. Cousin

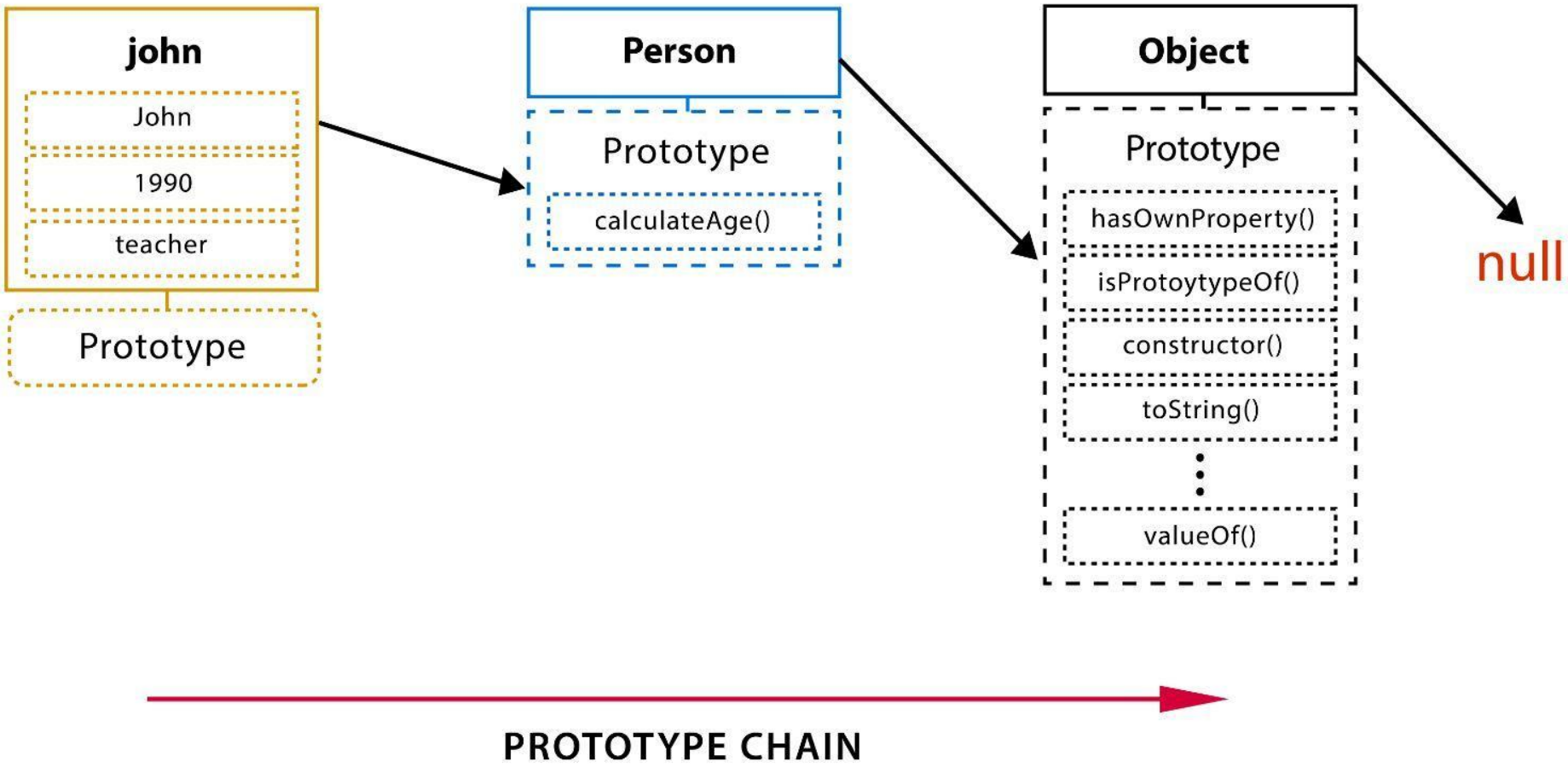


Super class VS Sub class



Prototypical inheritance refers to the ability to access object properties from another object. We use a JavaScript prototype to add new properties and methods to an existing object constructor.




john
John
1990
teacher
Prototype
Person
Prototype
calculateAge()
Object
Prototype
hasOwnProperty()
isProtoytypeOf()
constructor()
toString()
valueOf()
null
PROTOTYPE CHAIN

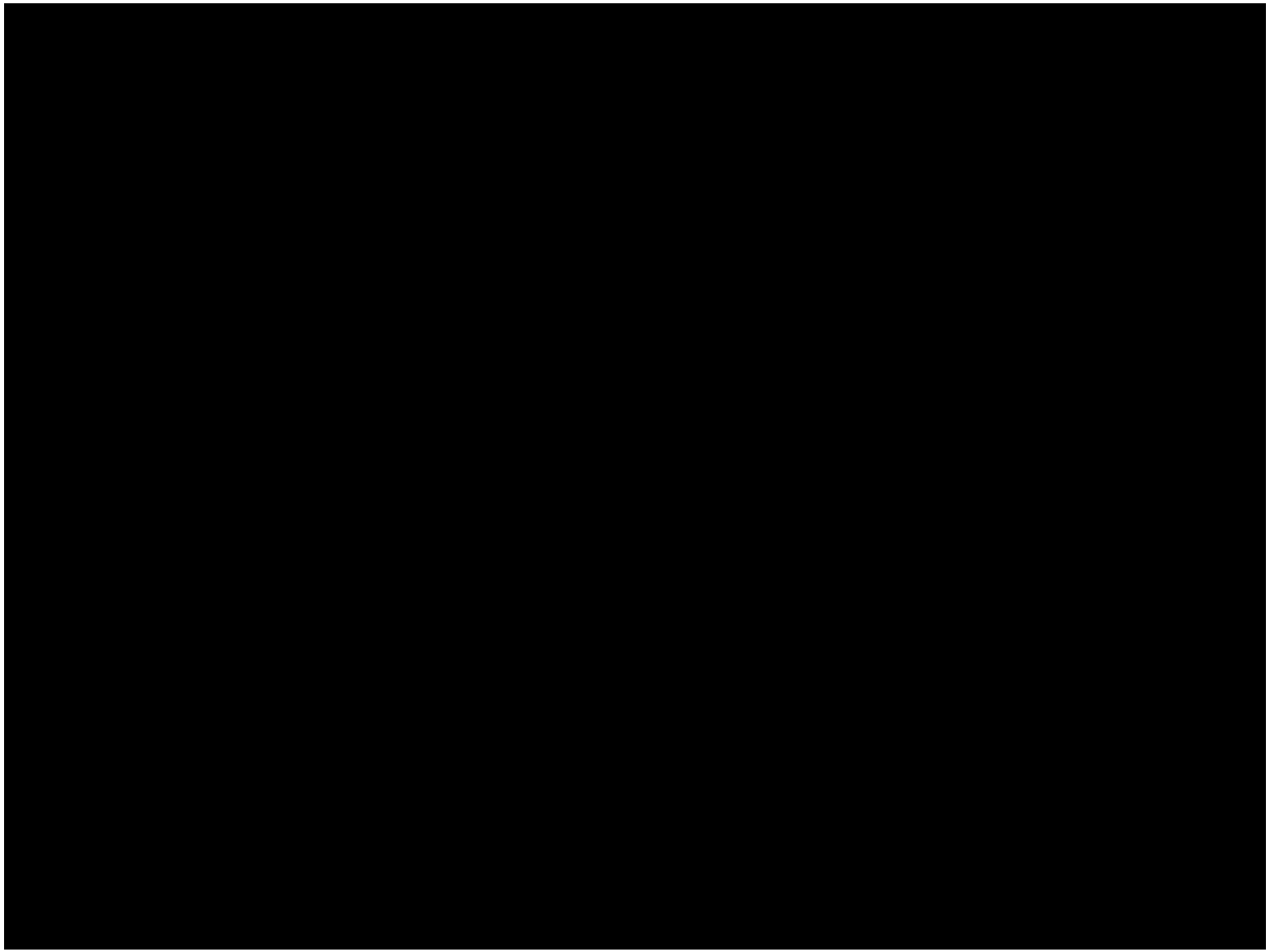
The Prototype Chain

Prototypal inheritance uses the concept of prototype chaining. Let's explore that concept. Every object created contains [[Prototype]], which points either to another object or null. Envision an object C with a [[Prototype]] property that points to object B. Object B's [[Prototype]] property points to prototype object A. This continues onward, forming a kind of chain called the prototype chain.

This concept is used when searching our code. When we need to find a property in an object, it is first searched for in the object, and if not found, it is searched for on that object's prototype, and so on. Thus, the entire prototype chain is traversed until the property is found or null is reached.

Ref: educative.io

The Prototype Inheritance, Prototype Chain



Ref: Victor Dozal