Array Methods

- forEach
- map
- filter
- some
- every
- reduce

forEach

```
arr.forEach(some function definition or name);
```

forEach

```
let arr = [1, 2, 3, 4, 5];

function print(el) {
    console.log(el);
}

arr.forEach(print);

// OR

arr.forEach(function(el) {
    console.log(el);
});
```

JS app.js > arr.forEach() callback

```
1   let arr = [1, 2, 3, 4, 5];
2
3   arr.forEach(function (el) {
4     console.log(el);
5   });
6
7   // let print = function (el) {
8   //   console.log(el);
9   // };
10
11  // arr.forEach(print);
12
```

No Issues

1	app.js:4
2	app.js:4
3	app.js:4
4	app.js:4
5	app.js:4

```
arr.forEach((el) => {
  console.log(el);
});
```

```
let arr = [
  {
    name: "aman",
    marks: 95,
  },
  {
    name: "shradha",
    marks: 94.4,
  },
  {
    name: "rajat",
    marks: 92,
  },
];

arr.forEach((student) => {
  console.log(student.marks);
});
```

No Issues

95

94.4

92

Map

let newArr = arr.map(some function definition or name);

```
let num = [1, 2, 3, 4];

let double = num.map(function(el) {
    return el*2;
});
```

No Issues

```
> double
<· ▶ (4) [2, 4, 6, 8]
>
```

```
let num = [1, 2, 3, 4];

let double = num.map((el) => {
  return el * el;
});
```

```
(4) [1, 4, 9, 16]
```

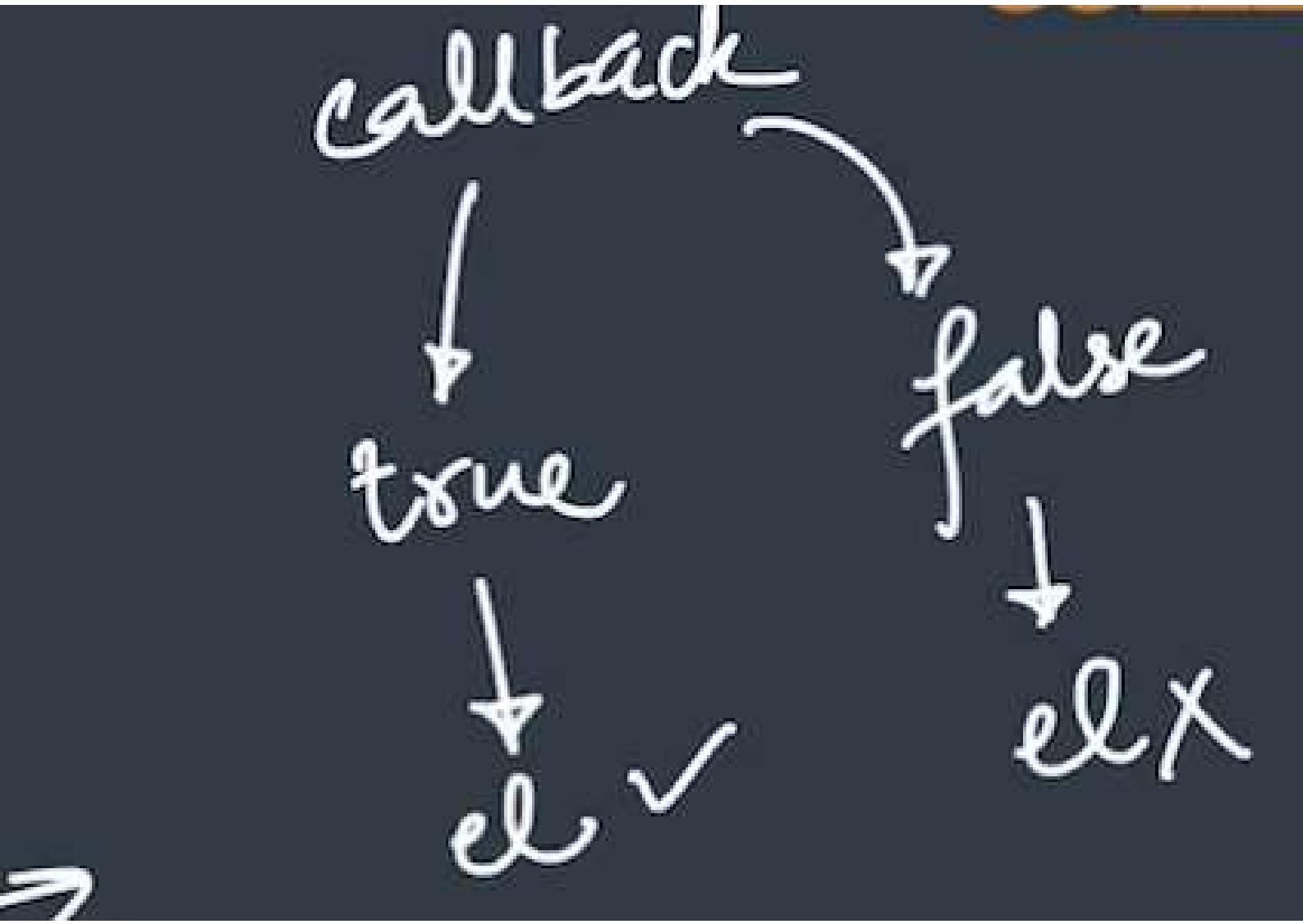
JS app.js > gpa > students.map() callback

```js
let students = [
  {
    name: "aman",
    marks: 95,
  },
  {
    name: "shradha",
    marks: 94.4,
  },
  {
    name: "rajat",
    marks: 92,
  },
];

let gpa = students.map((el) => {
  return marks / 10;
});
```

Filter

let newArr = arr.filter(some function definition or name);

```
let nums = [2, 4, 1, 5, 6, 2, 7, 8, 9];

let even = nums.filter( (num) => (num % 2 == 0) );
```


callback
true
false
el ✓
el ✗

```
let nums = [1, 2, 3, 4, 7, 8, 2, 9, 10, 12, 11];
let ans = nums.filter((el) => {
  return el % 2 == 0; //even -> true, odd -> false
});
```

No Issues

```
> ans
<· ▶ (6) [2, 4, 8, 2, 10, 12]
>
```

```
let nums = [1, 2, 3, 4, 7, 8, 2, 9, 10, 12, 11];
let ans = nums.filter((el) => {
  return el < 5;
});
```

No Issues

```
> ans
<· ▶ (5) [1, 2, 3, 4, 2]
>
```

Every

true

false

Returns true if every element of array gives true for some function. Else returns false.

arr.every(some function definition or name);

true false

```
[1, 2, 3, 4].every( (el) => (el%2 == 0));
false
[2, 4].every( (el) => (el%2 == 0));
true
```

No Issues

```
> [2, 4, 6].every((el) => el%2 === 0);
<· true
```

```
> [2, 4, 6, 8, 1].every((el) => el%2 == 0);
false
>
```

Some

T

Returns true if some elements of array give true for some function. Else returns false.

arr.some(some function definition or name);

```
[1, 2, 3, 4].some( (el) => (el%2 == 0));
true
[1, 3].some( (el) => (el%2 == 0));
false
```

Reduce

Reduces the array to a single value

arr.reduce(reducer function with 2 variables for (accumulator, element));


```
[1, 2, 3, 4].reduce( (res, el) => (res+el) );
10
```

[1, 2, 3]

① acc, el → return

② acc, el → return

③ acc, el → return

Reduced

```
let nums = [1, 2, 3, 4];
let finalVal = nums.reduce((res, el) => res + el);
console.log(finalVal);
```

No Issues

10

>

```
let nums = [1, 2, 3, 4];
let finalVal = nums.reduce((res, el) => {
  console.log(res);
  res + el;
});
console.log(finalVal);
```

No Issues

1

3

6

10

>

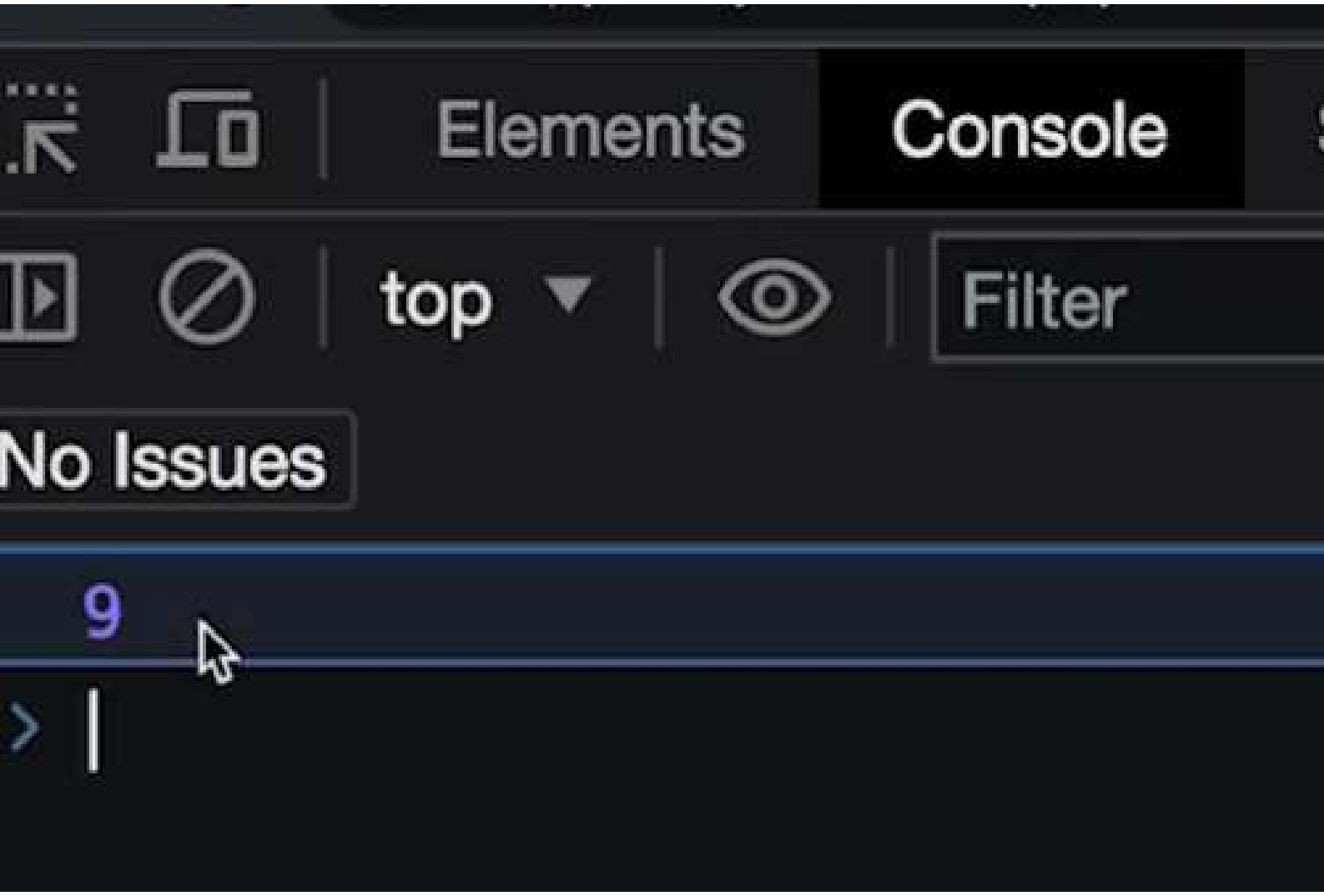
Reduce

Finding Maximum in an array

```
let nums = [2, 3, 4, 5, 3, 4, 7, 8, 1, 2];

let result = nums.reduce( (max, el) => {
    if(el > max) {
        return el;
    } else {
        return max;
    }
});
```

```
let max = arr.reduce((max, el) => {
  if (max < el) {
    return el;
  } else {
    return max;
  }
});

console.log(max)
```

snehagupta73


Elements
Console
top
Filter
No Issues
9

[1, 2, 3, 1] (max, el)

(0, 1) ⇒ 1

(1, 2) ⇒ 2

(2, 3) ⇒ 3

(3, 1) ⇒ 3

```
7, 8, 1, 2];
```



```
el) => {
```

Practice Qs

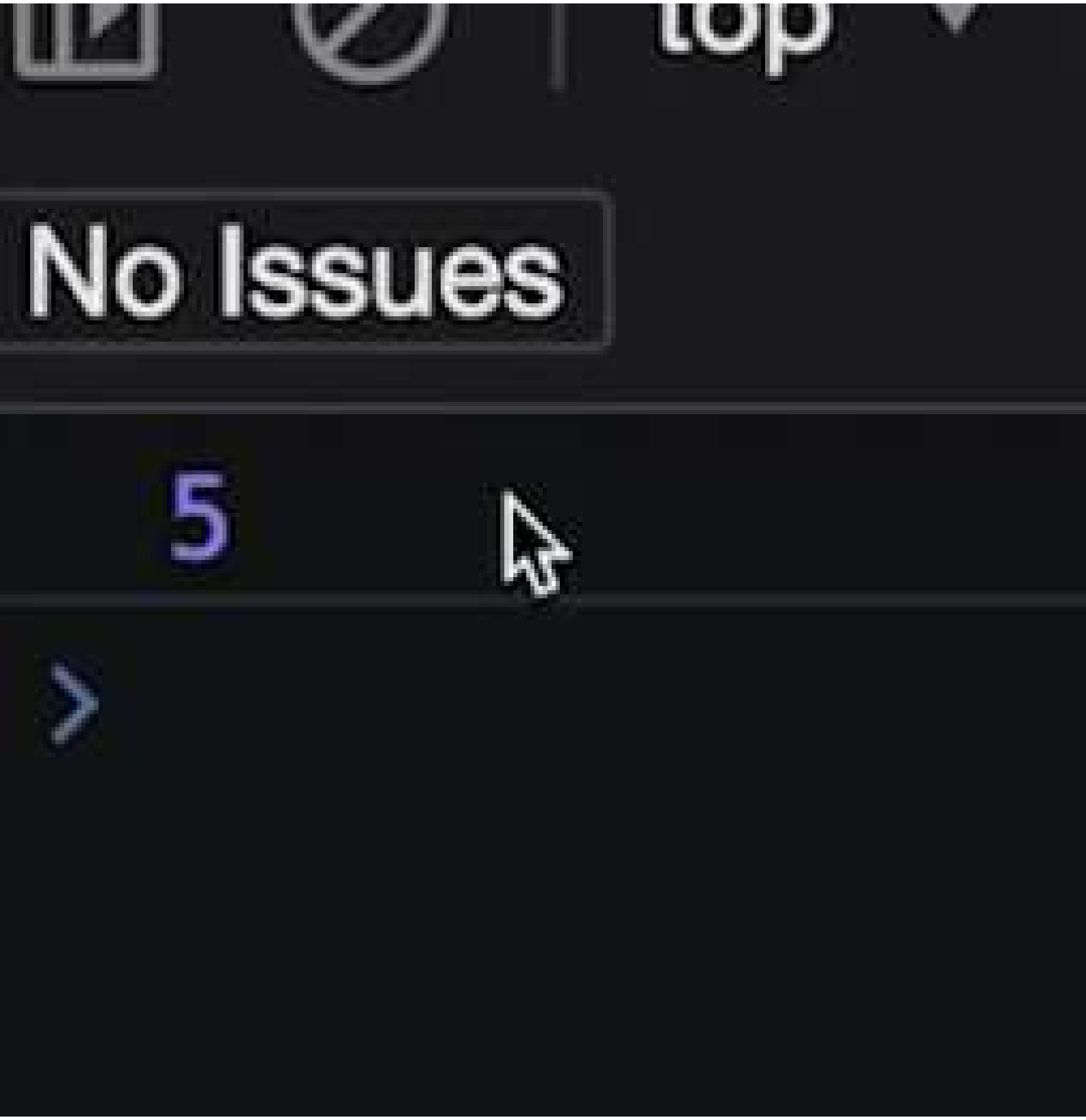
Check if all numbers in our array are multiples of 10 or not.

Create a function to find the min number in an array.

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

let ans = nums.every((el) => el%10 == 0)
```

JS app.js > nums

```js
let nums = [10, 20, 30, 40, 5];

let min = nums.reduce((min, el) => {
  if (min < el) {
    return min;
  } else {
    return el;
  }
});

console.log(min);
```


top
No Issues
5

Default Parameters

Giving a default value to the arguments

```
function func (a, b = 2) {
    //do something
}
```

```
function sum(a, b = 3) {
  return a + b;
}

sum(2); //5
```

Every

Returns true if every element of array gives true for some function. Else returns false.

arr.every(some function definition or name);

```
[1, 2, 3, 4].every( (el) => (el%2 == 0));
false
```

```
[2, 4].every( (el) => (el%2 == 0));
true
```

```
function sum(a = 2, b) {
  return a + b;
}


sum(1, 3); //4
sum(1); //a = 1, b = undefined
```

```
> sum(1, 3); //4
  sum(1); //a = 1, b = undefined
<· NaN
```

```
> sum(1, 3); //4
<· 4
```

```
> sum(1, 3); //4
  sum(1); //a = 1, b = undefined
<· NaN
```

Spread

Expands an iterable into multiple values

```
function func (...arr) {
   //do something
}
```

```
> let arr = [1, 2, 3, 4, 5];
<· undefined
> Math.min(...arr);
<· 1
> console.log(...arr);
  1 2 3 4 5
```

```
> console.log(..."apnacollege");
  a p n a c o l l e g e
```

```
> Math
<· ▶ Math {abs: ƒ, acos: ƒ, acosh: ƒ, asin: ƒ, asinh: ƒ, …}
> Math.min
<· ƒ min() { [native code] }
> Math.min(1, 2, 3);
<· 1
> Math.min(1, 2, 3, 1, 2, 3, 0);
<· 0
> Math.min(1, 2, 3, 1, 2, 3, 0, 1, 2, 3, 1, 2, 3, 0);
<· 0
> let arr = [1, 2, 3, 1, 2, 3, 0, 1, 2, 3, 1, 2, 3, 0];
<· undefined
> arr
<· ▶ (14) [1, 2, 3, 1, 2, 3, 0, 1, 2, 3, 1, 2, 3, 0]
> Math.min(...arr);
<· 0
>
```

```
> console.log(...arr);
  1 2 3 1 2 3 0 1 2 3 1 2 3 0 -1
<· undefined
> console.log(1, 2, 3, 1);
  1 2 3 1
<· undefined
```

```
> console.log(..."apnacollege");
  a p n a c o l l e g e
```

Spread

with Array Literals

```
> let arr = [1, 2, 3, 4, 5];
<· undefined
> let newArr = [...arr];
<· undefined
> newArr
<· ▶ (5) [1, 2, 3, 4, 5]
```

```
> let chars = [..."hello"];
<· undefined
> chars
<· ▶ (5) ['h', 'e', 'l', 'l', 'o']
```

```
let arr = [1, 2, 3, 4, 5];
let newArr = [...arr];
```

```
> arr
<· ▶ (5) [1, 2, 3, 4, 5]
> newArr
<· ▶ (5) [1, 2, 3, 4, 5]
>
```

```
let chars = [    "hello"]
```

```
> chars
: ▶ (5) ['h', 'e', 'l', 'l', 'o']
```

```
let odd = [1, 3, 5, 7, 9];
let even = [2, 4, 6, 8, 10];
```



```
let nums = [...odd, ...even];
```

```
> nums
<· ▶ (10) [1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
>
```

```
let odd = [1, 3, 5, 7, 9];
let even = [2, 4, 6, 8, 10];

let nums = [...even, ...odd];
```

Spread

with Object Literals

```
let data = {
  email: "ironman@gmail.com",
  password: "abcd",
};

let dataCopy = { ...data, id: 123 };
```

```
const data = {
  email: "ironman@gmail.com",
  password: "abcd",
};

const dataCopy = { ...data, id: 123 };
```

```
> dataCopy
<· ▶ {email: 'ironman@gmail.com', password: 'abcd', id: 123}
```

```
let arr = [1, 2, 3, 4, 5]; //val
let obj1 = { ...arr }; //obj -> key:val
```

```
> obj1
<· ▼ {0: 1, 1: 2, 2: 3, 3: 4, 4: 5}
    0: 1
    1: 2
    2: 3
    3: 4
    4: 5
  ▶ [[Prototype]]: Object
```

```
let obj2 = { ...hello" };
```

Rest ↔

Allows a function to take an indefinite number of arguments and bundle them in an array

```
function sum(...args) {
  return args.reduce((add, el) => add + el);
}
```

Rest

Allows a function to take an indefinite number of arguments and bundle them in an array

```
function sum(...args) {
  return args.reduce((add, el) => add + el);
}
```

```
function sum(...args) {
  //arguments
  for (let i = 0; i < args.length; i++) {
    console.log("you gave us: ", args[i]);
  }
}
```

```
> sum(1);
  you gave us:  1
<· undefined
> sum(1, 2);
  you gave us:  1
  you gave us:  2
<· undefined
> sum(1, 2, 3, 4);
  you gave us:  1
  you gave us:  2
  you gave us:  3
  you gave us:  4
<· undefined
>
```

```
function min(a, b, c, d) {
  console.log(arguments);
}
```

```
> min(1, 2, 3, 4);
  ▶ Arguments(4) [1, 2, 3, 4, callee: ƒ, Symbol(Symbol.iterator): ƒ]
<· undefined
>
```

```
function min() {
  console.log(arguments);
  console.log(arguments.length);
  arguments.push(1);
}
```

```
> min(1, 2, 3, 4);
  ▶ Arguments(4) [1, 2, 3, 4, callee: ƒ, Symbol(Symbol.iterator): ƒ]
  4
  ▶ Uncaught TypeError: arguments.push is not a function
      at min (app.js:11:13)
      at <anonymous>:1:1
>
```

```
function sum() {
  return arguments.reduce((sum, el) => sum + el);
}
```

```
> sum(1, 2, 3, 4);
Uncaught TypeError: arguments.reduce is not a function
    at sum (app.js:2:20)
    at <anonymous>:1:1
>
```

```
function sum(...args) {
  return args.reduce((sum, el) => sum + el);
}
```

```
> sum(1, 2, 3, 4);
< 10
```

```
function min(...args) {
    return args.reduce((min, el) => {
        if(min > el) {
            return el;
        } else {
            return min;
        }
    })
}
```

```
> min(1, 2, 3, 4);
<· 1
```

```
function min(msg, ...args) {
  console.log(msg);
  return args.reduce((min, el) => {
    if (min > el) {
      return el;
    } else {
      return min;
    }
  });
}
```

```
> min("hello",12, 445, 123, -20);
  hello
<· -20
```

Destructuring

Storing values of array into multiple variables

```
let names = ["tony", "bruce", "steve", "peter"];
let [winner, runnerup] = names;
console.log(winner, runnerup); //"tony" "bruce"
```

```
let names = ["tony", "bruce", "peter", "steve"];
// let winner = names[0];
// let runnerup = names[1];
// let secondRunnerup = names[2];

let [winner, runnerup, secondRunnerup] = names;
```

```
> winner
<· 'tony'
> runnerup
<· 'bruce'
>
```

```
obj2
▼ {0: 'h', 1: 'e', 2: 'l', 3: 'l', 4: 'o'}
    0: "h"
    1: "e"
    2: "l"
    3: "l"
    4: "o"
  ▶ [[Prototype]]: Object
```

```
> others
<· ▶ (5) ['peter', 'steve', 'abc', 'xyz', 'pyq']
```

Destructuring

Objects

```
const student = {
  name: "karan",
  class: 9,
  age: 14,
  subjects: ["hindi", "english", "math", "science", "social studies"],
  username: "karan123",
  password: 1234,
};

const { username: user, password: pass } = student;

console.log(user); //"karan123"
```

```
const student = {
  name: "karan",
  age: 14,
  class: 9,
  subjects: ["hindi", "english", "math", "science"],
  username: "karan@123",
  password: "abcd",
};

let { username, password } = student;
```

```
> username
<· 'karan@123'
> password
<· 'abcd'
```

```
const student = {
  name: "karan",
  age: 14,
  class: 9,
  subjects: ["hindi", "english", "math", "science"],
  username: "karan@123",
  password: "abcd",
};

let { username: user, password: secret, city = "Mumbai" } = student;
```