CENTRO UNIVERSITARIO
DE TECNOLOGÍA
Y ARTE DIGITAL

Arrays and Objects

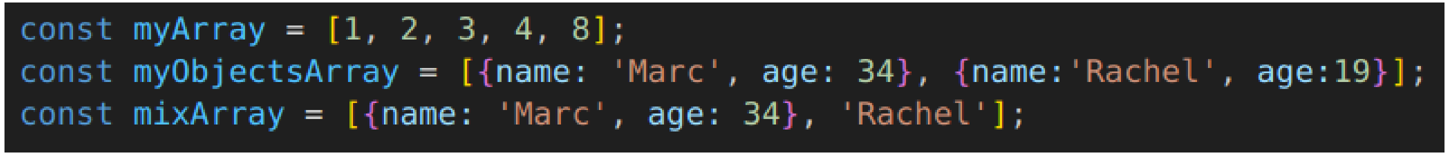
2024-2025

CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

Contents

Arrays

- Array is a set of elements indexed of any type, including objects.
- Different elements of the same array could have diferent types.
- They are resizable



```
const myArray = [1, 2, 3, 4, 8];
const myObjectsArray = [{name: 'Marc', age: 34}, {name:'Rachel', age:19}];
const mixArray = [{name: 'Marc', age: 34}, 'Rachel'];
```

Arrays (Try it)

- Define an array with 6 elements.
- Try to read the 10th element
- Try to fill the 10th position with any value.
- Show the array on the console.

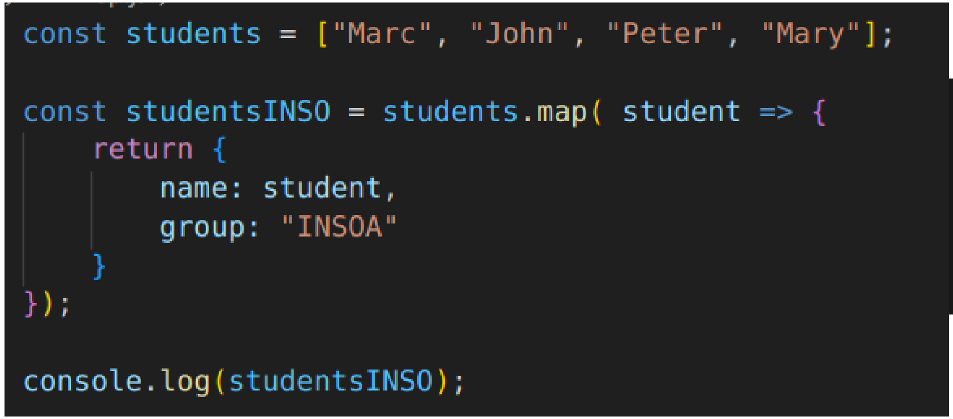
Iterative methods

- Many array methods take a callback function as an argument
- The callback function is called sequentially and at most once for each element in the array, and the return value of the callback function is used to determine the return value of the method
- The callback function could take 3 arguments:
  - Item
  - Index
  - Array

  Most of the times index and array are not necessary.
- These functions doesn´t modify the array.

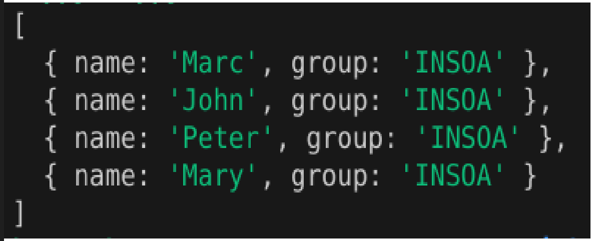
Iterative methods (map)

- creates a new array populated with the results of calling a provided function on every element in the calling array



```
const students = ["Marc", "John", "Peter", "Mary"];

const studentsINSO = students.map( student => {
    return {
        name: student,
        group: "INSOA"
    }
});

console.log(studentsINSO);
```

Output:

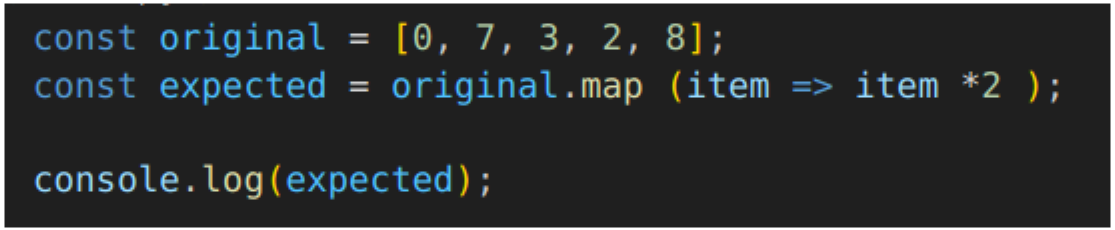


```
[
  { name: 'Marc', group: 'INSOA' },
  { name: 'John', group: 'INSOA' },
  { name: 'Peter', group: 'INSOA' },
  { name: 'Mary', group: 'INSOA' }
]
```

Iterative methods (map)

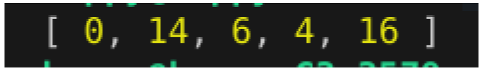
- Use the map metod to duplicate the value of the array items
- Example:
  - Original array: [0, 7, 3, 2, 8];
  - Array expected: [0, 14, 6, 4, 16]

Iterative methods (map)



```
const original = [0, 7, 3, 2, 8];
const expected = original.map (item => item *2 );

console.log(expected);
```

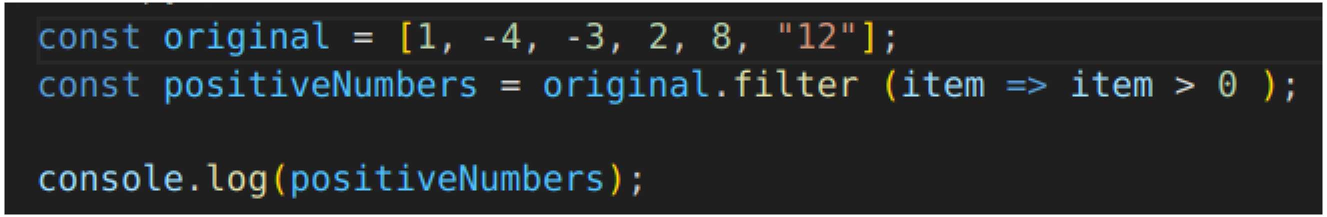
Output:



```
[ 0, 14, 6, 4, 16 ]
```

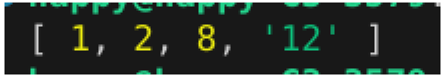
Iterative methods (filter)

- This method creates a shallow copy of a portion of a given array, filtered down to just the elements from the given array that pass the test implemented by the provided function.



```
const original = [1, -4, -3, 2, 8, "12"];
const positiveNumbers = original.filter (item => item > 0 );

console.log(positiveNumbers);
```

Output:



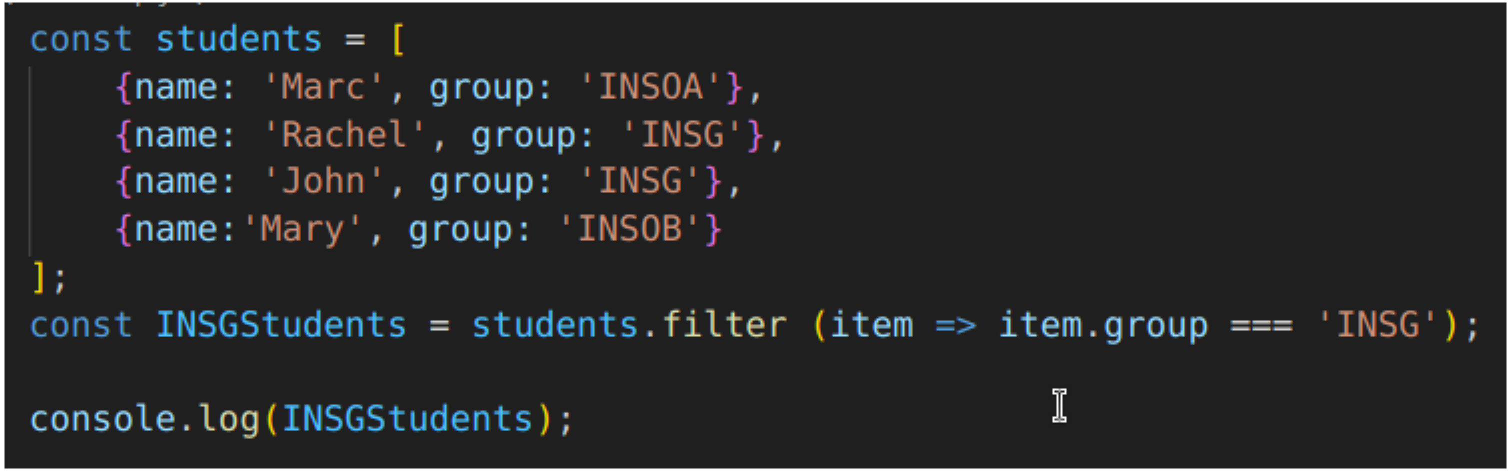
```
[ 1, 2, 8, '12' ]
```

Iterative methods (filter)

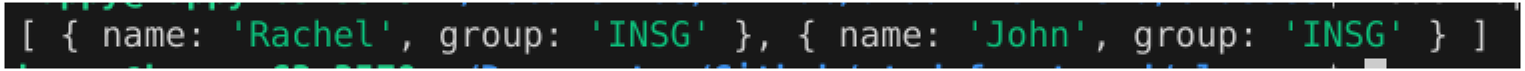
- Use the filter metod to obtain the elements with the group 'INSG'

- Example:
  - Original array: [{name: 'Marc', group: 'INSOA'}, {name: 'Rachel', group: 'INSG'}, {name: 'John', group: 'INSG'}, {name:'Mary', group: 'INSOB'}]
  - Array expected: [{name: 'Rachel', group: 'INSG'}, {name: 'John', group: 'INSG'}]

Iterative methods (filter)



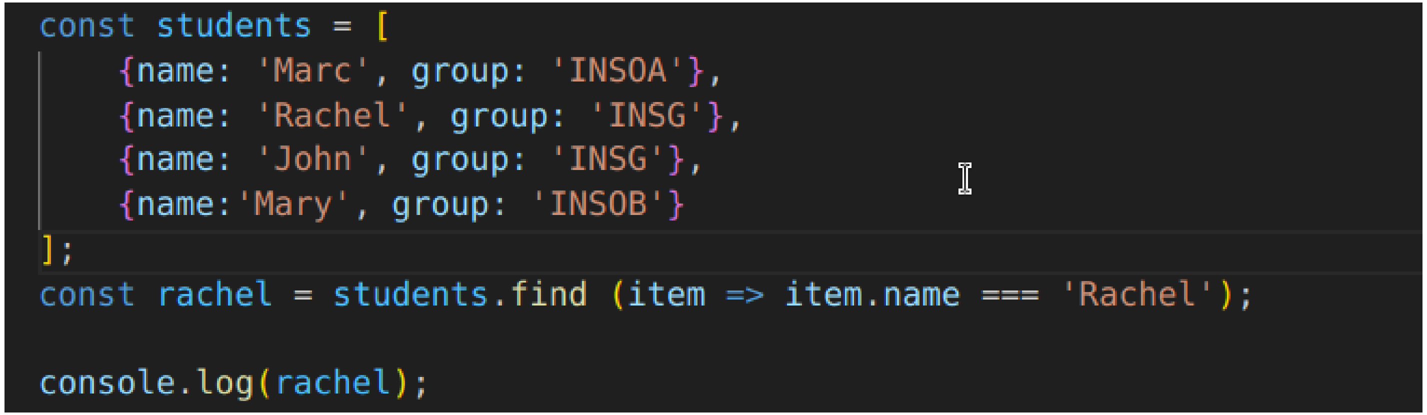
```
const students = [
    {name: 'Marc', group: 'INSOA'},
    {name: 'Rachel', group: 'INSG'},
    {name: 'John', group: 'INSG'},
    {name:'Mary', group: 'INSOB'}
];
const INSGStudents = students.filter (item => item.group === 'INSG');

console.log(INSGStudents);
```



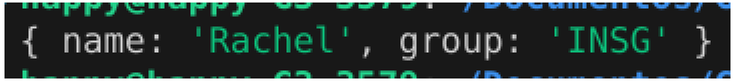
```
[ { name: 'Rachel', group: 'INSG' }, { name: 'John', group: 'INSG' } ]
```

Iterative methods (find)

- This method returns the first element in the provided array that satisfies the provided testing function.



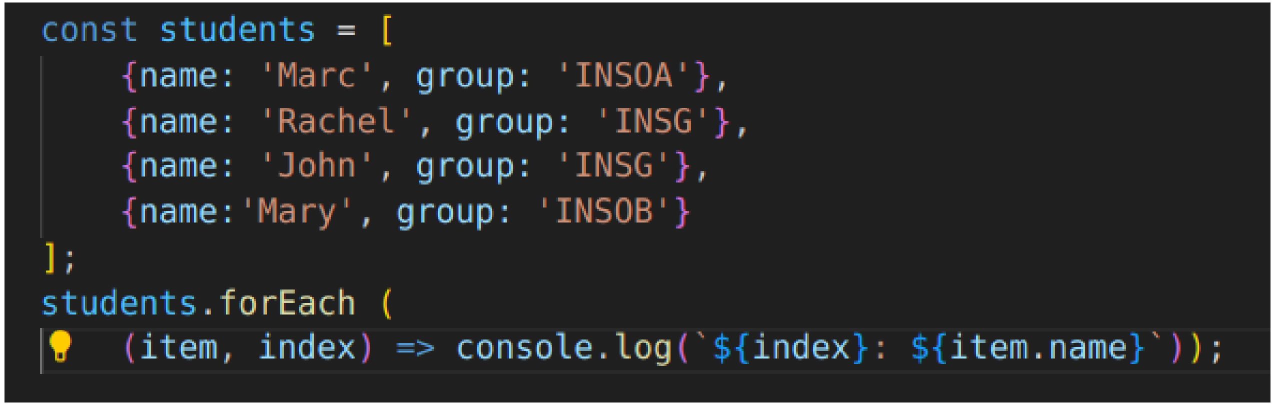
```
const students = [
    {name: 'Marc', group: 'INSOA'},
    {name: 'Rachel', group: 'INSG'},
    {name: 'John', group: 'INSG'},
    {name:'Mary', group: 'INSOB'}
];
const rachel = students.find (item => item.name === 'Rachel');

console.log(rachel);
```



```
{ name: 'Rachel', group: 'INSG' }
```

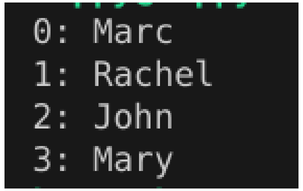
Iterative methods (forEach)

- This method executes a provided function once for each array element. It returns undefined.



```
const students = [
    {name: 'Marc', group: 'INSOA'},
    {name: 'Rachel', group: 'INSG'},
    {name: 'John', group: 'INSG'},
    {name:'Mary', group: 'INSOB'}
];
students.forEach (
    (item, index) => console.log(`${index}: ${item.name}`));
```

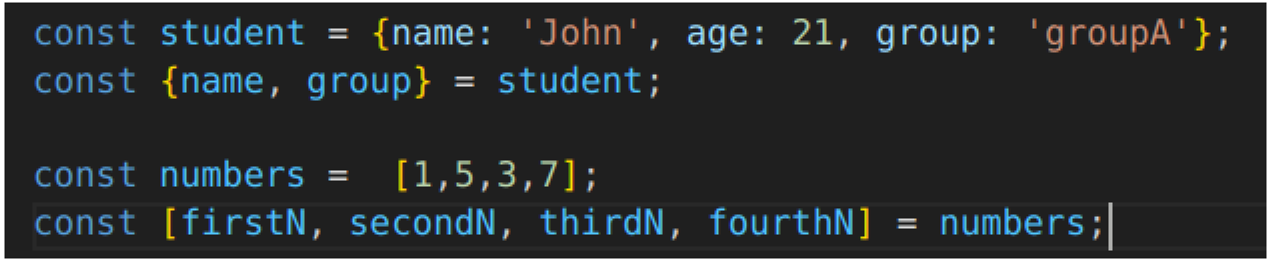
Output:



```
0: Marc
1: Rachel
2: John
3: Mary
```

Destructuring syntax

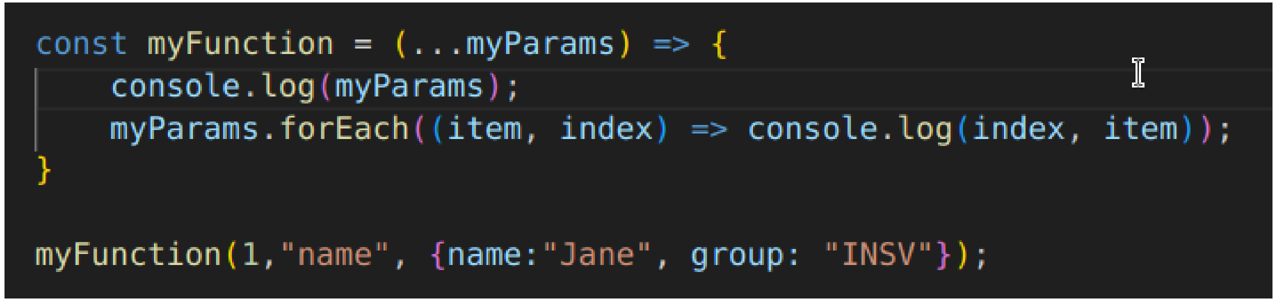
- The **destructuring assignment** syntax is a JavaScript expression that makes it possible to unpack
    - values from arrays,
    - properties from objects.



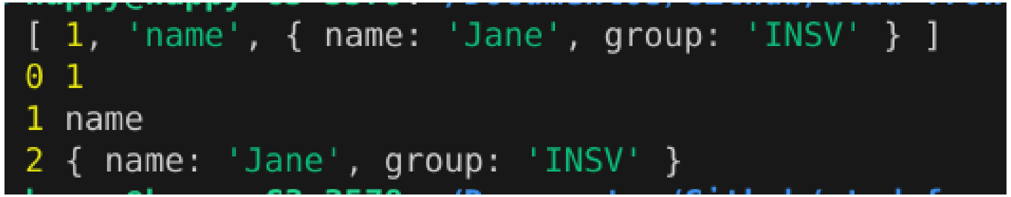
```
const student = {name: 'John', age: 21, group: 'groupA'};
const {name, group} = student;

const numbers =  [1,5,3,7];
const [firstN, secondN, thirdN, fourthN] = numbers;
```

Rest parameter

- The **rest parameter** syntax allows a function to accept an indefinite number of arguments as an array.



```javascript
const myFunction = (...myParams) => {
    console.log(myParams);
    myParams.forEach((item, index) => console.log(index, item));
}

myFunction(1,"name", {name:"Jane", group: "INSV"});
```



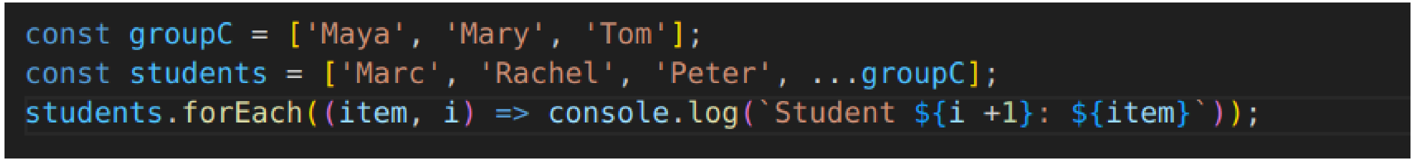
```
[ 1, 'name', { name: 'Jane', group: 'INSV' } ]
0 1
1 name
2 { name: 'Jane', group: 'INSV' }
```

Rest parameter

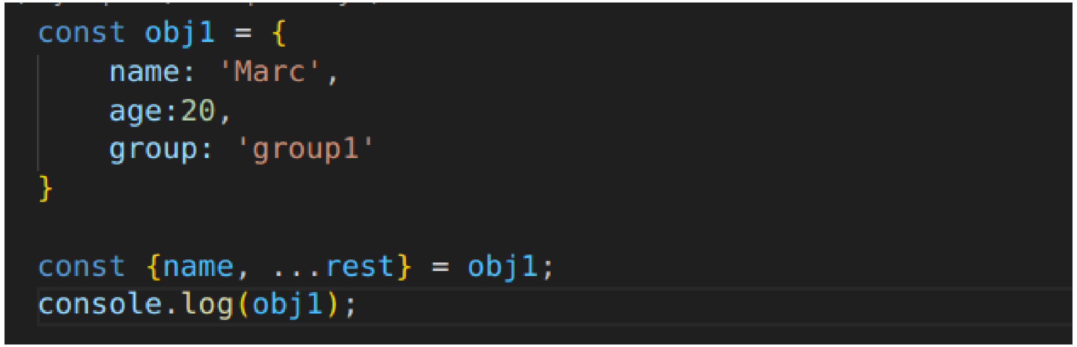
- **Restriccions:**
  - A function definition can only have one rest parameter.
  - The rest parameter must be the last parameter in the function definition.
  - The rest parameter cannot have a default value.

Spread operator

The **spread (...)** syntax allows an iterable, such as an array or string, to be expanded in places where zero or more arguments (for function calls) or elements (for array literals) are expected.



```
const groupC = ['Maya', 'Mary', 'Tom'];
const students = ['Marc', 'Rachel', 'Peter', ...groupC];
students.forEach((item, i) => console.log(`Student ${i +1}: ${item}`));
```



```
const obj1 = {
    name: 'Marc',
    age:20,
    group: 'group1'
}

const {name, ...rest} = obj1;
console.log(obj1);
```

Spread operator

- Exercise 1:
    - Define an array with several numbers
    - Create a copy using the spread operator
    - Modify first and second arrays.
    - Print them on the console output.
- Exercise 2:
    - Define an array with several objects
    - Each object must have two properties: name and color
    - Add an object to the second array
    - Modify the names of any object.
    - Print them on the console output.

Exercises

1. Given an array of mixed types, filter out the even numbers, then multiply them by 2, and finally add all the resulting values.

2. Combine several arrays of numbers using the spread operator, delete duplicate ones, and then calculate the average of all the numbers.

3. We have an array of objects. Each object represents a book, with the properties title, author, year and price. Create a function that filters and sorts books by price and returns the selected books with title and author information. The function receives a comma-separated list of books as a parameter.