ES 6 Features

By CODEMIND Technology

Contact us 966 5044 698
966 5044 598

ES6 Features

- “use strict” Mode
- Template Literals or String Template
- Let and const
- Ternary Operator → ? :
- Spread operators (...) and Rest Parameter
- Classes
- Set and Map
- Function default parameter
- Self invoking function or IIFE
- JS Destructuring
- Arrow Function
- Promises
- map( ), filter( ), reduce( ) methods
- sort( ) and reverse( )

advance

JavaScript ES6 (also known as ECMAScript 2015 or ECMAScript 6) is the newer version of JavaScript that was introduced in 2015.

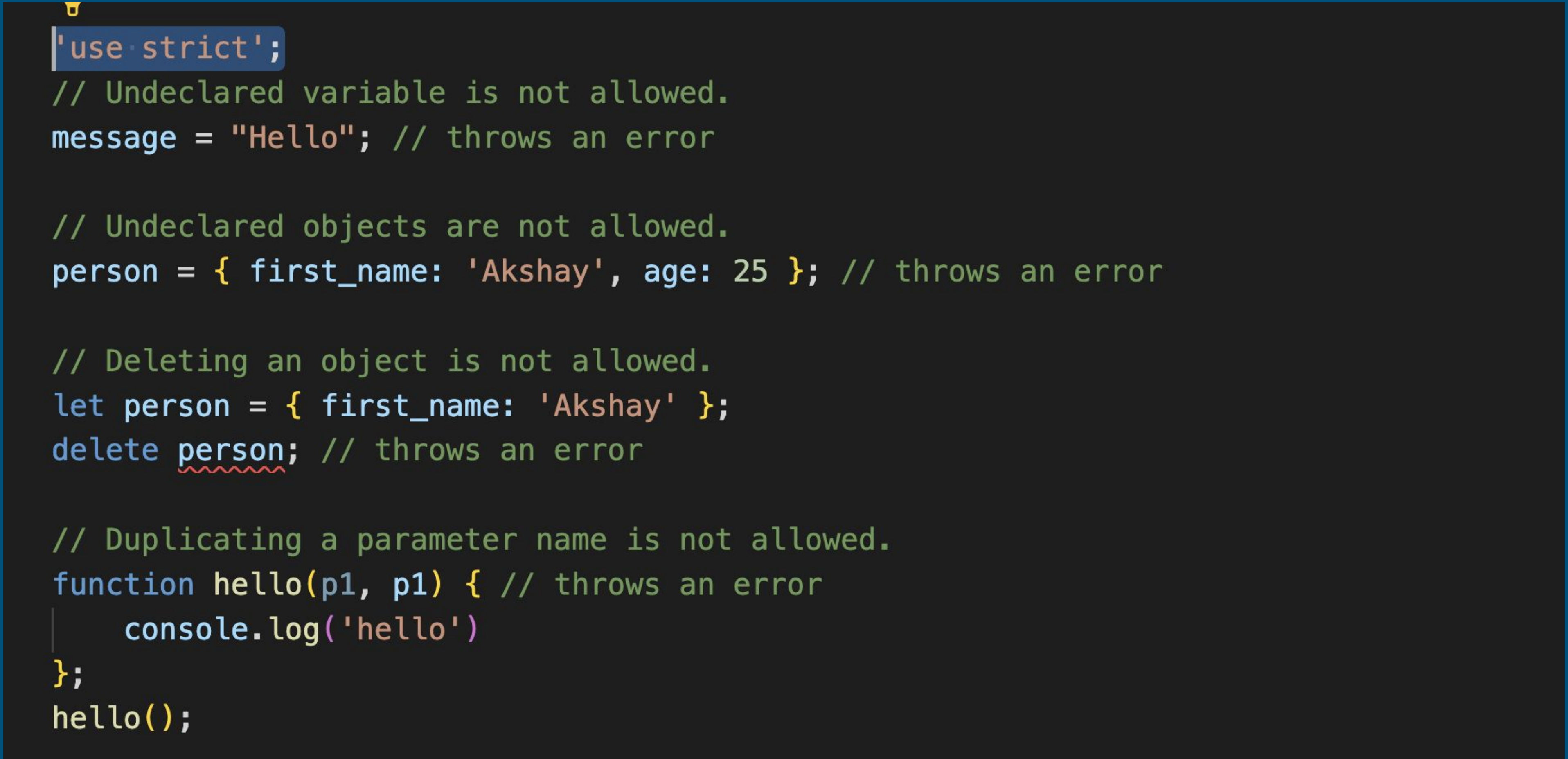
JavaScript “use strict”

In JavaScript, 'use strict'; states that the code should be executed in 'strict mode'. This makes it easier to write good and secure JS code

Benefits of Strict Mode

- Helps to write a cleaner code
- Changes previously accepted silent errors (bad syntax) into real errors and throws an error message
- Makes it easier to write "secure" JavaScript

JavaScript “use-strict”



```
'use strict';
// Undeclared variable is not allowed.
message = "Hello"; // throws an error

// Undeclared objects are not allowed.
person = { first_name: 'Akshay', age: 25 }; // throws an error

// Deleting an object is not allowed.
let person = { first_name: 'Akshay' };
delete person; // throws an error

// Duplicating a parameter name is not allowed.
function hello(p1, p1) { // throws an error
    console.log('hello')
};
hello();
```

String template - Template Literals

Use backticks (\` \`) characters to define string template.

Few usage of string template are:

1. In a string putting a word with double quote “ ”

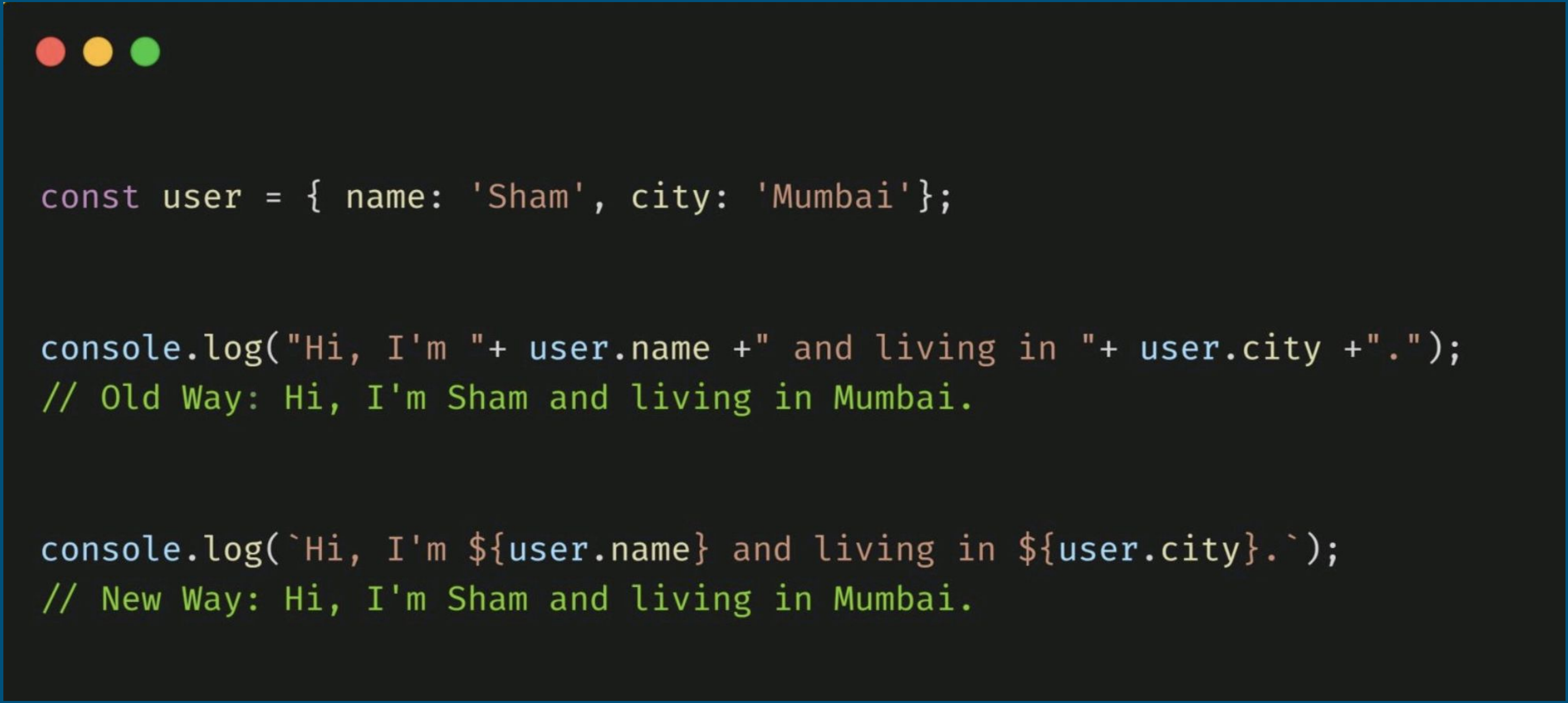
```
var message = `Hello, Very Good Morning - "Sachin" `;
console.log(message);
```

2. Variable Substitution

```
var firstName = ”Codemind”;
var lastName = “Technology”
console.log(`First Name: ${firstName} and Last Name: ${lastName}`);
```

Note: \`\` (backticks) is different than ‘ ’ (single quote)

String template - ES6 feature



```
const user = { name: 'Sham', city: 'Mumbai'};


console.log("Hi, I'm "+ user.name +" and living in "+ user.city +".");
// Old Way: Hi, I'm Sham and living in Mumbai.


console.log(`Hi, I'm ${user.name} and living in ${user.city}.`);
// New Way: Hi, I'm Sham and living in Mumbai.
```

Spread Operator: … (3 dots)

The spread operator ... is used to expand an array or spread the elements

```
const fruits = ["Apple", "Mango", "Orange", "Strawberry", "Grapes"];
console.log(fruits); // (5) ['Apple', 'Mango', 'Orange', 'Strawberry', 'Grapes']
console.log(...fruits); // Apple Mango Orange Strawberry Grapes
```

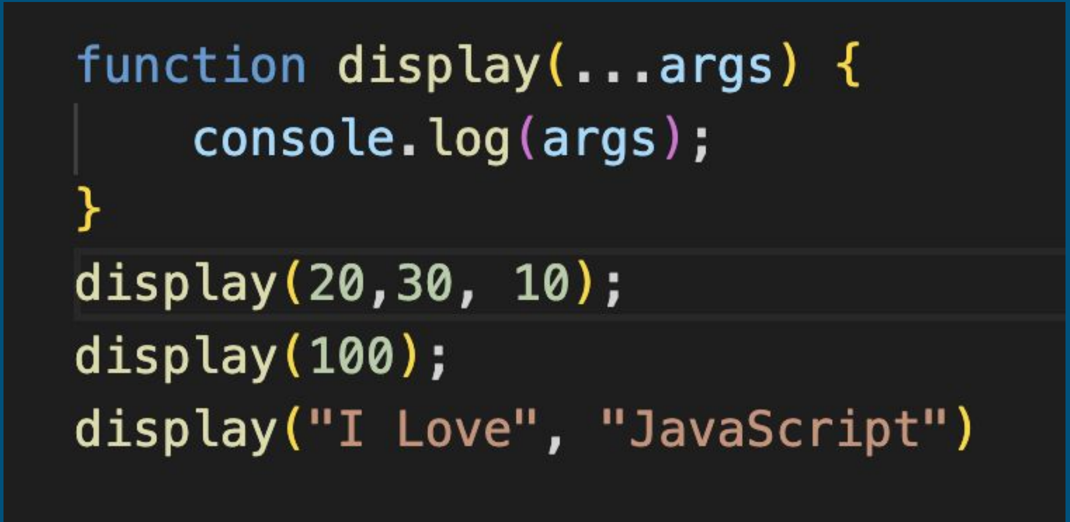
In this code: `console.log(...fruits);`

Is equivalent to: `console.log("Apple", "Mango", "Orange", "Strawberry", "Grapes");`

Rest Parameter

- When the spread operator (...) used as parameter in the function, it i/k/a Rest Parameter.
- We can accept multiple arguments in a function call using the rest parameter
- Rule: Rest parameter should be the last argument in a function

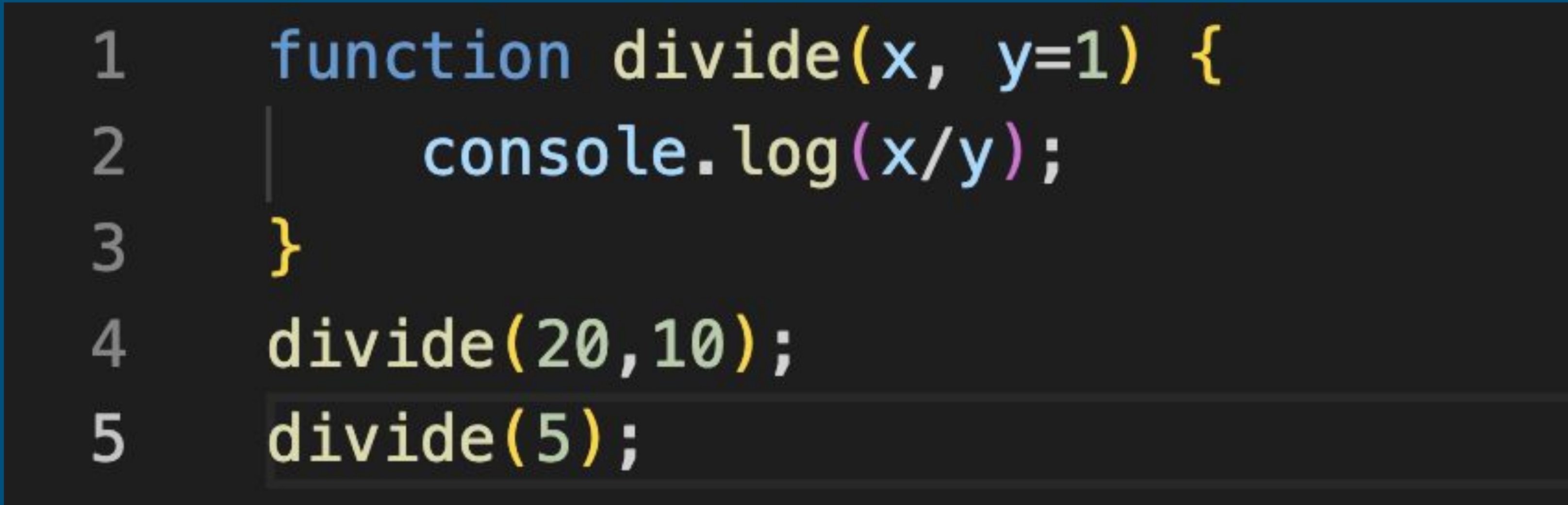
Note: Using the Rest parameter, will pass the arguments as array elements



```
function display(...args) {
    console.log(args);
}
display(20,30, 10);
display(100);
display("I Love", "JavaScript")
```

Function default parameters

- In JS, default function parameters allow you to initialize named parameters with default values if no values or undefined are passed into the function.
- As shown in below snippet. when we don’t pass the value for y parmeter, it will take 1 as default value inside divide( ) function invoked at line number → 5

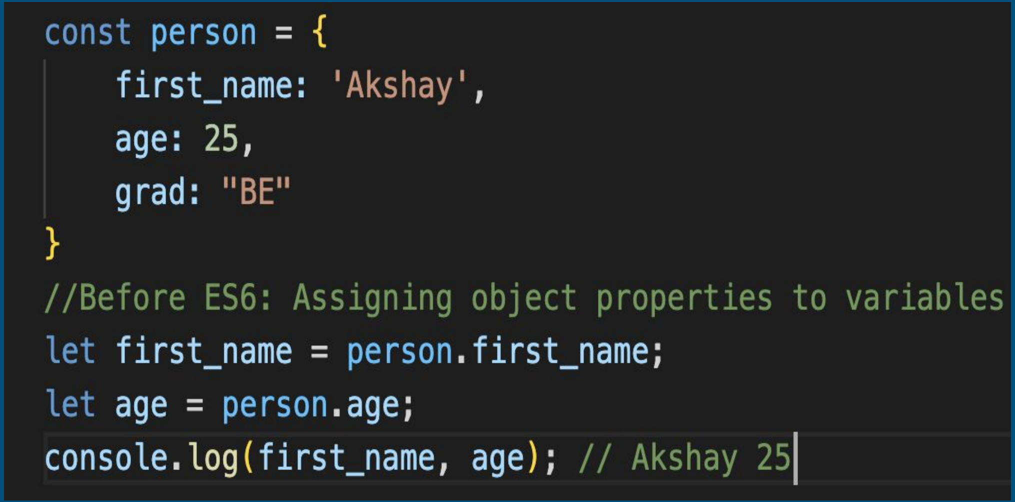


```
1    function divide(x, y=1) {
2        console.log(x/y);
3    }
4    divide(20,10);
5    divide(5);
```

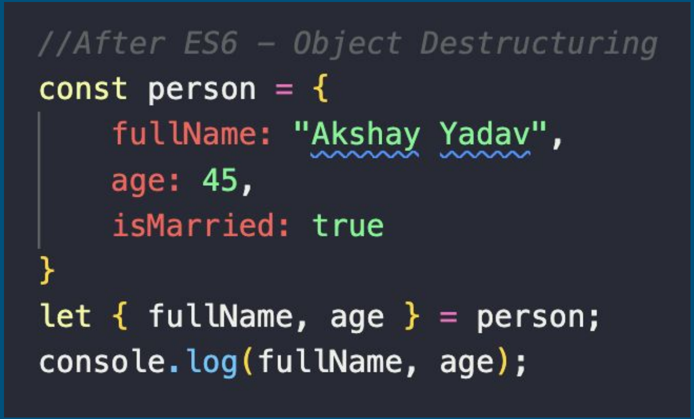
Destructuring: Object Destructuring, Array Destructuring

This destructuring ES6 features makes easy to extract properties from an object or an element from an array.

This feature will make easy to assign object properties to distinct variables.
Note: When destructuring objects, we should use the same name for the variable as the corresponding object properties.



```
const person = {
    first_name: 'Akshay',
    age: 25,
    grad: "BE"
}
//Before ES6: Assigning object properties to variables
let first_name = person.first_name;
let age = person.age;
console.log(first_name, age); // Akshay 25
```

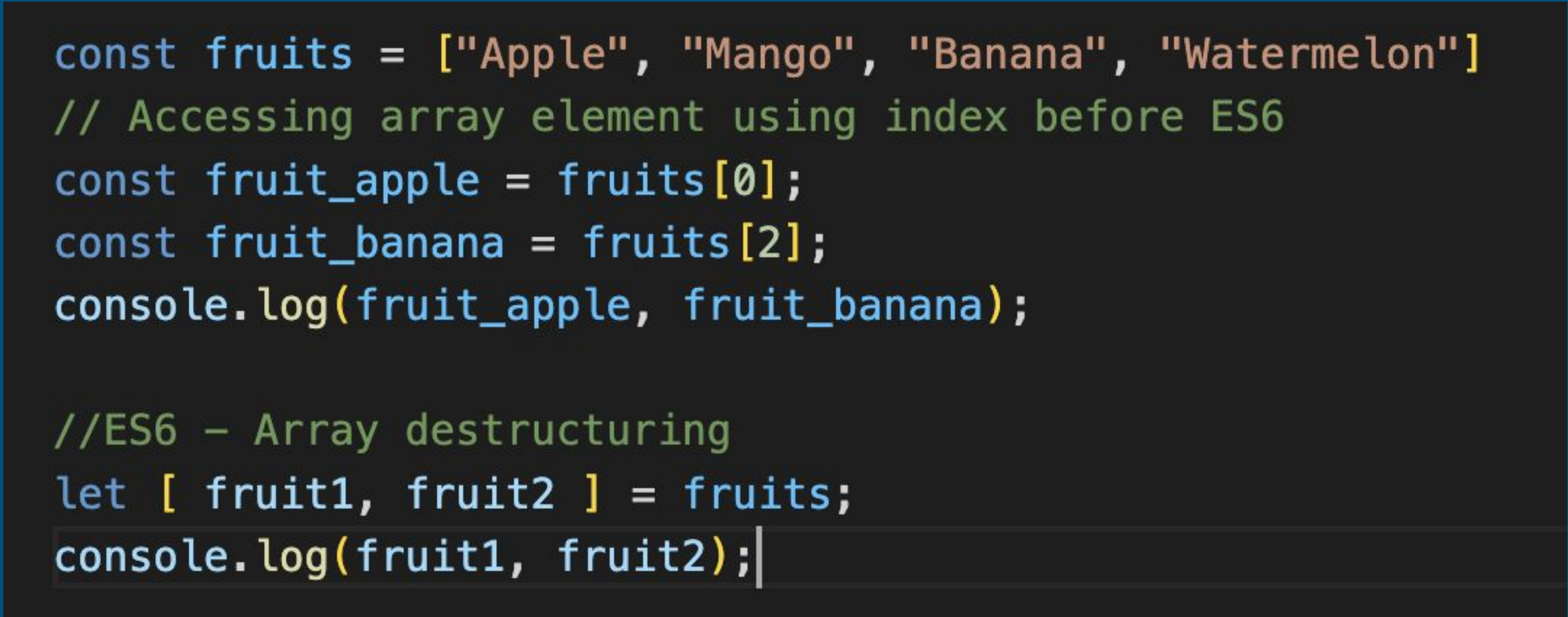


```
//After ES6 - Object Destructuring
const person = {
    fullName: "Akshay Yadav",
    age: 45,
    isMarried: true
}
let { fullName, age } = person;
console.log(fullName, age);
```

Array Destructuring

With the help of this feature we can extract array element into separate variable

Syntax → let [element1, element2, element3] = array_name

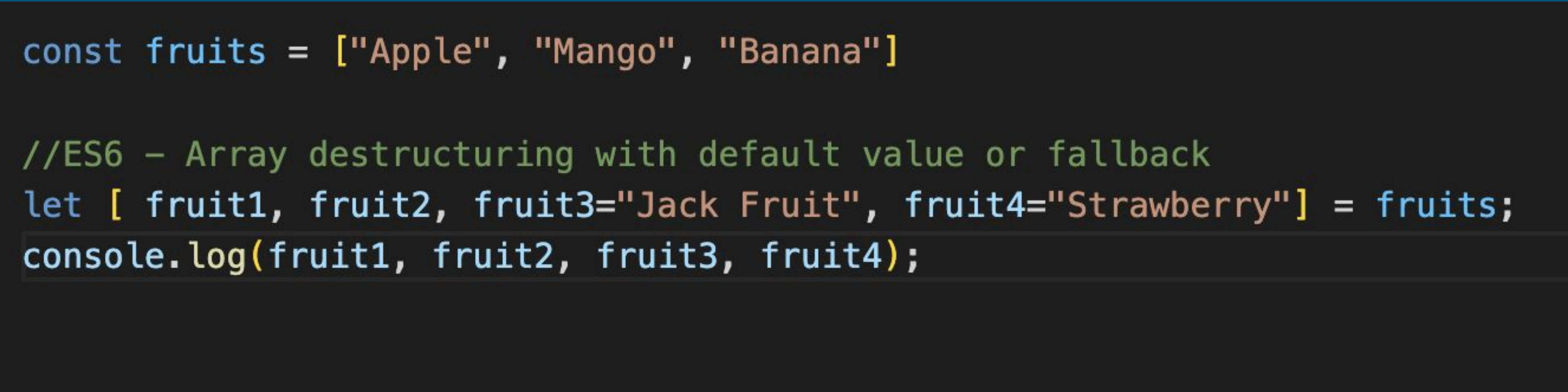


```
const fruits = ["Apple", "Mango", "Banana", "Watermelon"]
// Accessing array element using index before ES6
const fruit_apple = fruits[0];
const fruit_banana = fruits[2];
console.log(fruit_apple, fruit_banana);

//ES6 - Array destructuring
let [ fruit1, fruit2 ] = fruits;
console.log(fruit1, fruit2);
```

Array Destructuring with default values

Destructuring allows a default value to be assigned to a variable if no value or undefined is passed. It is like providing a fallback when nothing is found



```javascript
const fruits = ["Apple", "Mango", "Banana"]

//ES6 - Array destructuring with default value or fallback
let [ fruit1, fruit2, fruit3="Jack Fruit", fruit4="Strawberry"] = fruits;
console.log(fruit1, fruit2, fruit3, fruit4);
```

Assignment:

const arrayNum = [11, 3, 4, 11, 4, 7, 3];

Remove duplicate element from array

Given String value →

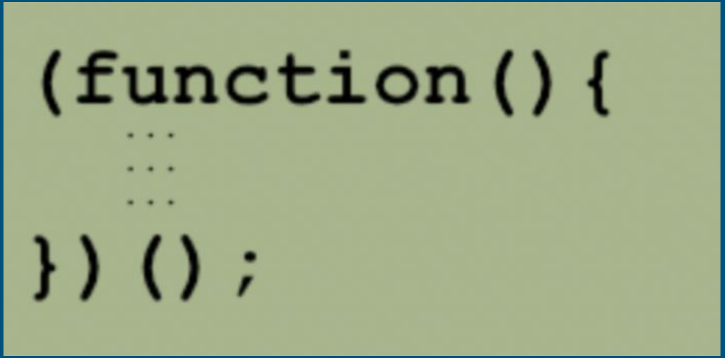
const str = "How are you mate";

Expected output ⇒ "HoW ArE YoU MatE"

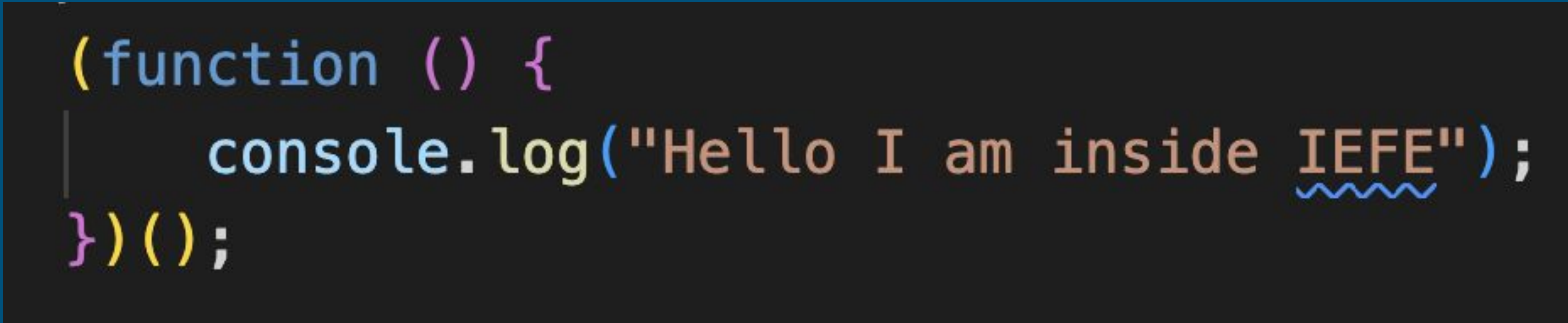
Self invoking Function or IIFE (Immediately Invoked Function Expression)

IIFE is a function defined as an expression and executed immediately after creation

Syntax:



```
(function(){
    ...
    ...
    ...
})();
```



```
(function () {
    console.log("Hello I am inside IEFE");
})();
```

Function Expression

Function which acts like a **value**...meaning you can intialize variable with function as a value.

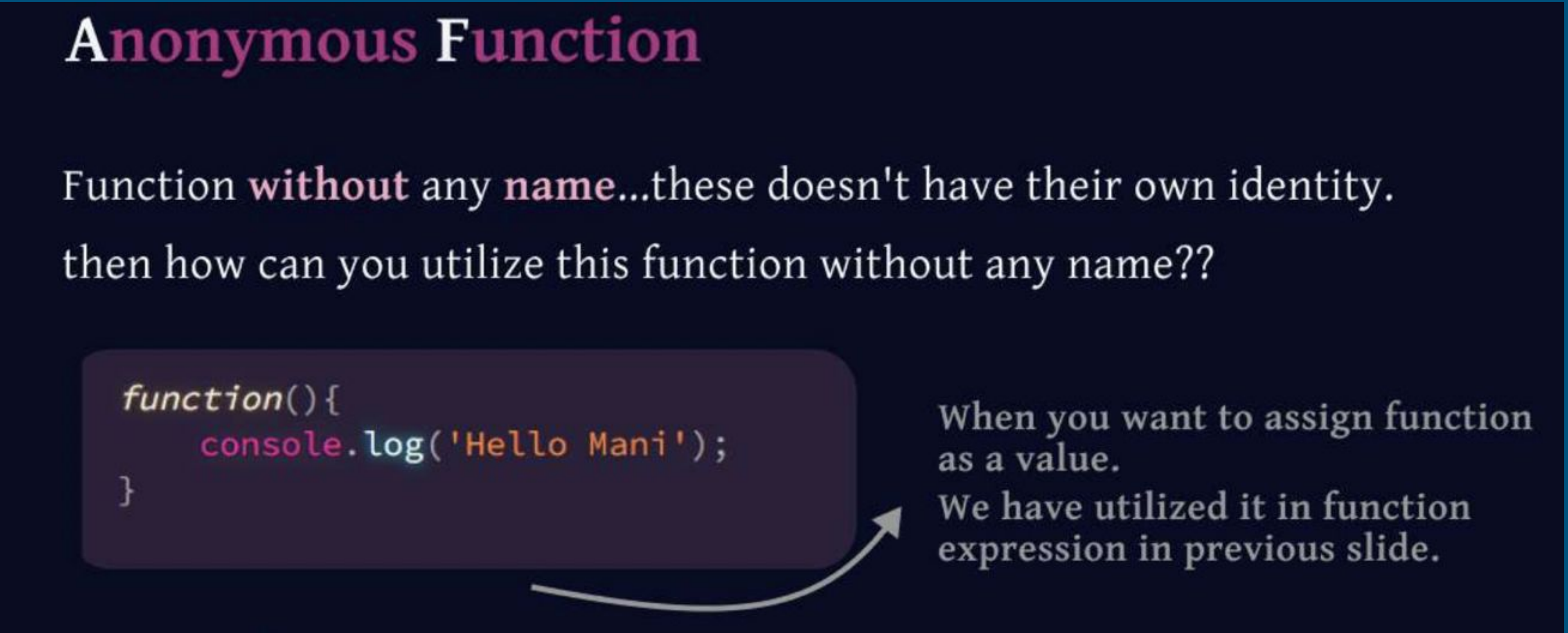
Function statement

```
function greet(){
  console.log('Hello User');
}
```

Function Expression

```
var greet = function (){
  console.log('Hello User');
}
```

Anonymous Function



Anonymous Function

Function **without** any **name**...these doesn't have their own identity.

then how can you utilize this function without any name??

```
function(){
    console.log('Hello Mani');
}
```

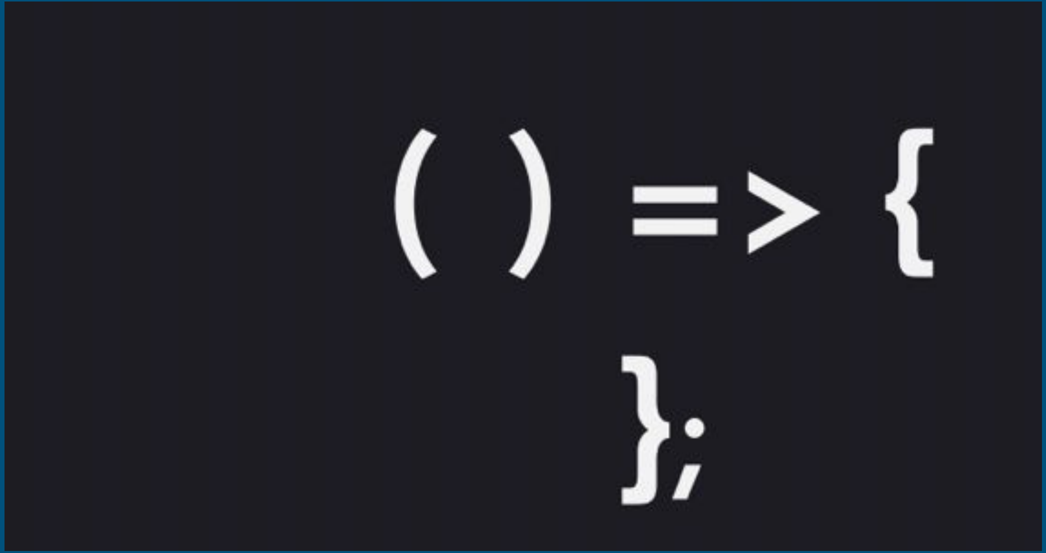
When you want to assign function as a value.
We have utilized it in function expression in previous slide.

Arrow Function

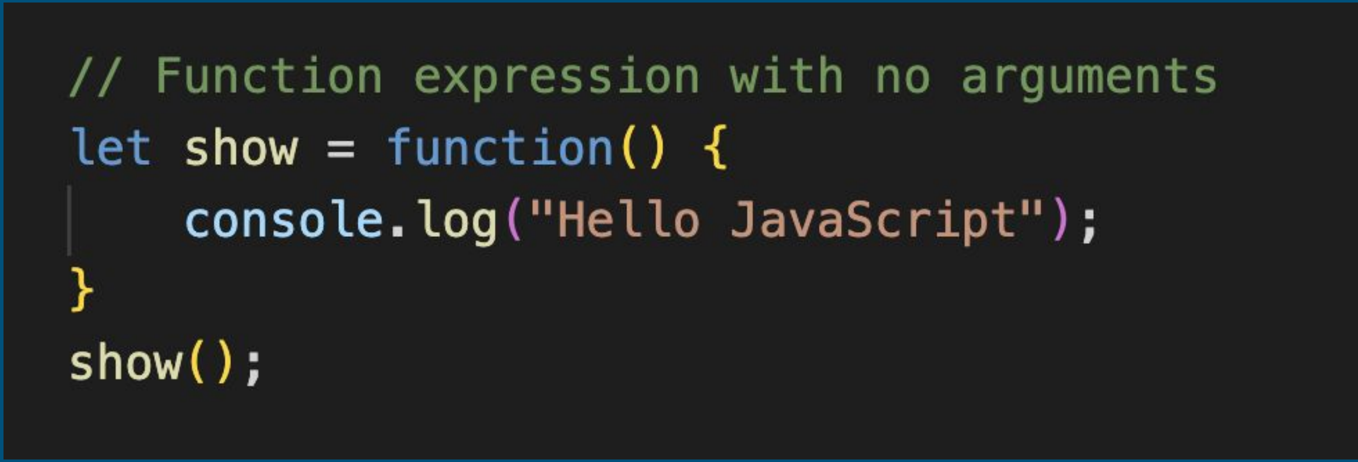
Arrow function is introduced in ES6, It allows us to create a functions in cleaner and shorter way as compared to regular functions.

Syntax:

```
 let fun = ( ) => {
   // statements
}
fun( );
```

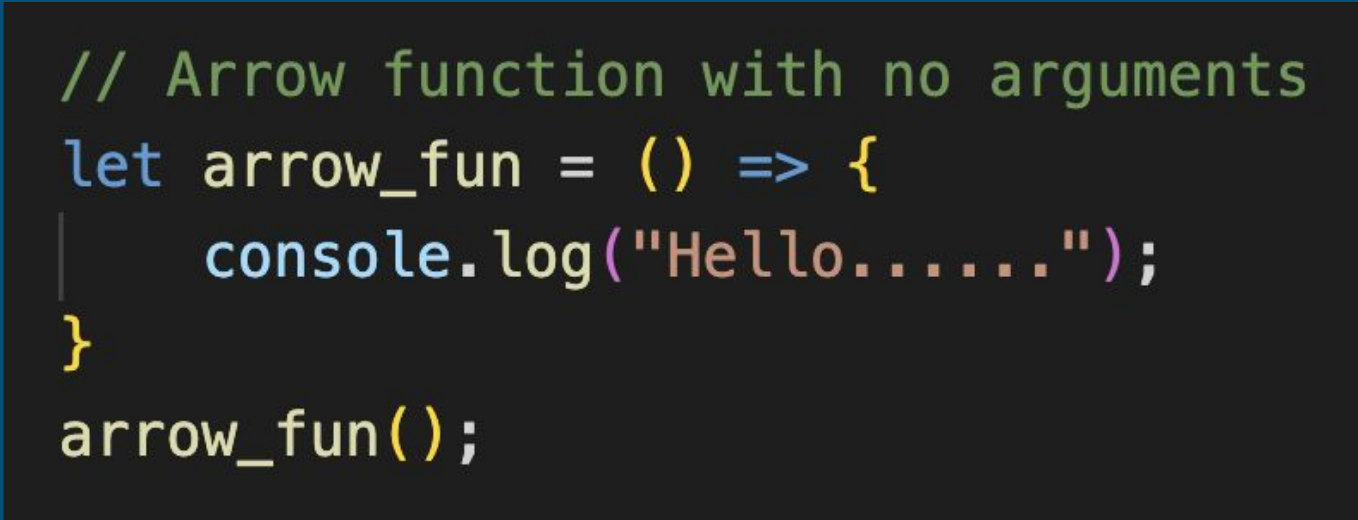

```
( ) => {
  };
```

Function Expression with no arguments



```
// Function expression with no arguments
let show = function() {
    console.log("Hello JavaScript");
}
show();
```

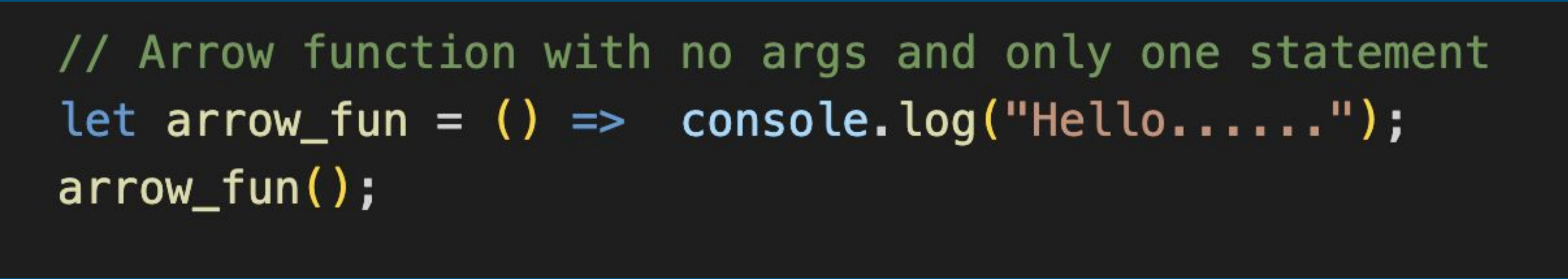
Arrow function with no arguments And only one stmts inside { }



```
// Arrow function with no arguments
let arrow_fun = () => {
    console.log("Hello......");
}
arrow_fun();
```

When there is only one statement inside arrow function we can also write like as below:

We can remove curly brackets



```
// Arrow function with no args and only one statement
let arrow_fun = () =>  console.log("Hello......");
arrow_fun();
```

Assignment 0A: Arrow function, 17_arrowFun_assigA.js

Write a arrow functions such that - Pls don’t forget to log result on console with meaningful msg

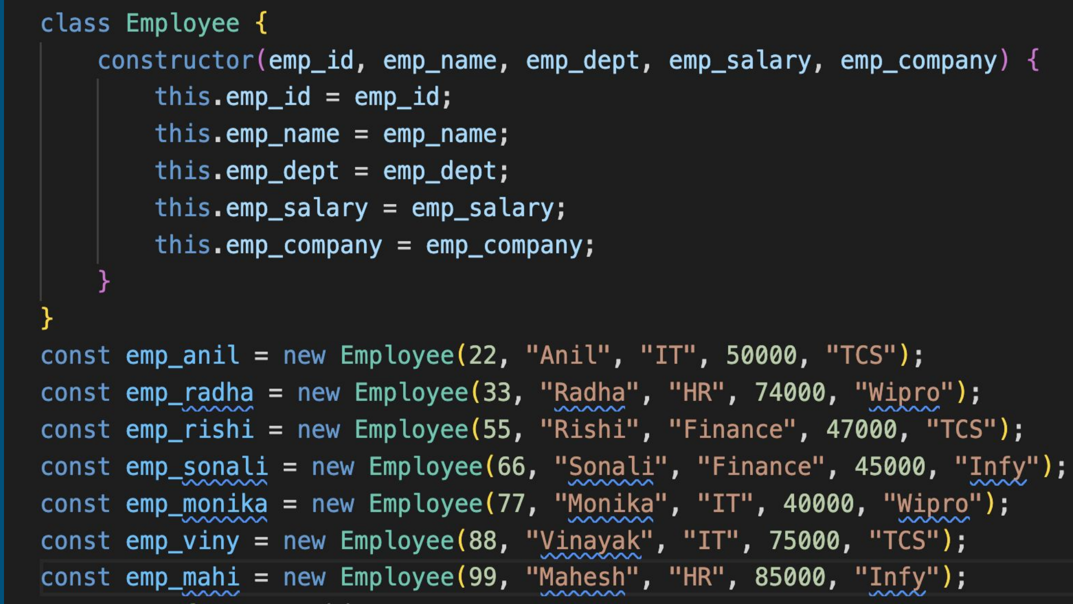
1. With no args and no return value, log message on console inside arrow function
    a. “Good Morning, Today is Monday”
2. With 3 args and no return value, for received 3 parameters perform the multiplication
    a. Values to be passed ⇒ 5, 5, 2
    b. Invoke the same function for values ⇒ 10, 4 [Note: assign default value to 3rd arg as 1]
3. With 5 args and return value such as, for received params it should do the addition
    a. Values to be passed ⇒ 100, 100, 200, 349, 756
    b. Log the returned result on console with meaningful message for both step 3.a and 3.c
    c. Invoke the same arrow function for values: “I am”, “ learning”, “ ES6”, ‘ features’, “ in depth”

Assignment 0B, 17_AssignB.js

Note: Please create the Employee class as it →
- While creating objects pass the values as it is →

- Add all the created emp objects inside array namely as ‘array_employess’

Solve the below problem statements as
& log result on console with meaningful msg



```
class Employee {
    constructor(emp_id, emp_name, emp_dept, emp_salary, emp_company) {
        this.emp_id = emp_id;
        this.emp_name = emp_name;
        this.emp_dept = emp_dept;
        this.emp_salary = emp_salary;
        this.emp_company = emp_company;
    }
}
const emp_anil = new Employee(22, "Anil", "IT", 50000, "TCS");
const emp_radha = new Employee(33, "Radha", "HR", 74000, "Wipro");
const emp_rishi = new Employee(55, "Rishi", "Finance", 47000, "TCS");
const emp_sonali = new Employee(66, "Sonali", "Finance", 45000, "Infy");
const emp_monika = new Employee(77, "Monika", "IT", 40000, "Wipro");
const emp_viny = new Employee(88, "Vinayak", "IT", 75000, "TCS");
const emp_mahi = new Employee(99, "Mahesh", "HR", 85000, "Infy");
```

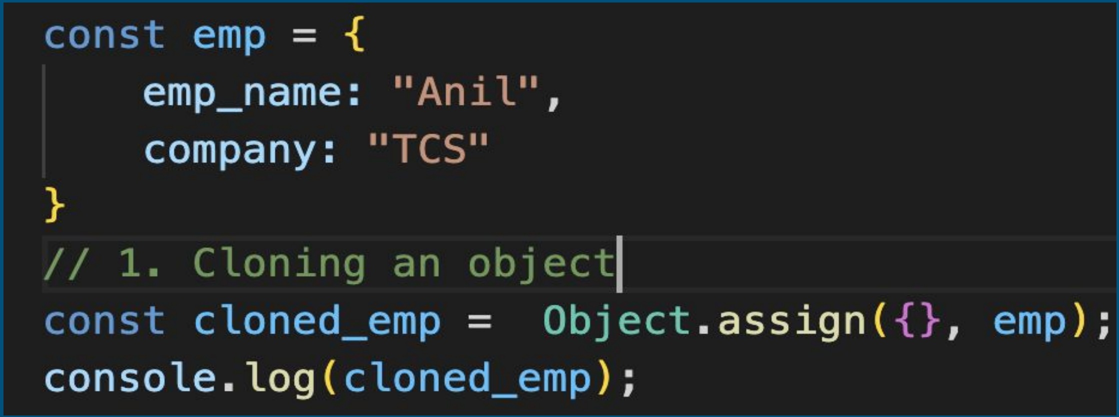
1. Find all the employees working in ‘TCS’ and log only employee <u>names and company </u>name on console
2. Find the ‘Finance’ department employees, log only <u>department and employee name</u> on console
3. Find the employees whose name starts with <u>‘R’ </u>and complete employee details on console [hint –> startsWith(“R“)]
4. Find the employees whose salary is greater than 75000, and log on console <u>emp name, company and salary</u>
5. Find the emp’s whose salary greater than or equal 50000 and from ‘IT’ department, log complete emp details on console
[Hint → if(emp.emp_salary>=50000 && emp.emp_dept==’IT’) inside for loop]

Object.assign( ):

Syntax: Object.assign(target, ...sources);

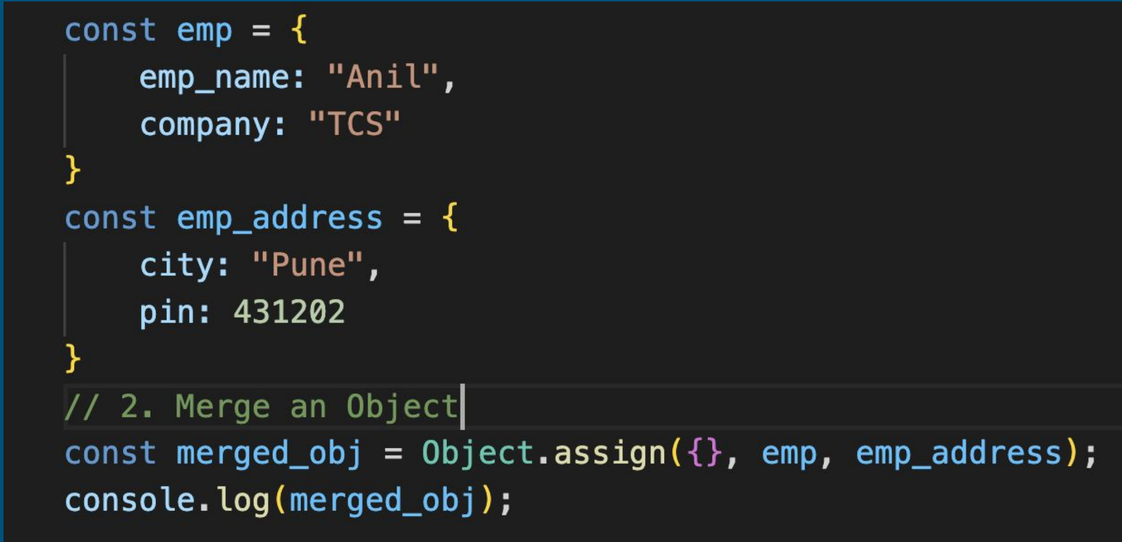
This method is used for

1. To Clone an object



```
const emp = {
    emp_name: "Anil",
    company: "TCS"
}
// 1. Cloning an object
const cloned_emp =  Object.assign({}, emp);
console.log(cloned_emp);
```

2. To merge an objects



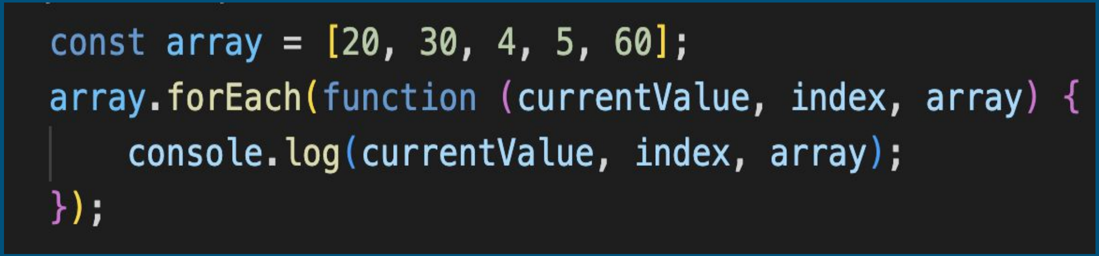
```
const emp = {
    emp_name: "Anil",
    company: "TCS"
}
const emp_address = {
    city: "Pune",
    pin: 431202
}
// 2. Merge an Object
const merged_obj = Object.assign({}, emp, emp_address);
console.log(merged_obj);
```

forEach( ) method or forEach loop

The forEach( ) passes a callback function for each element of an array together with the below parameters

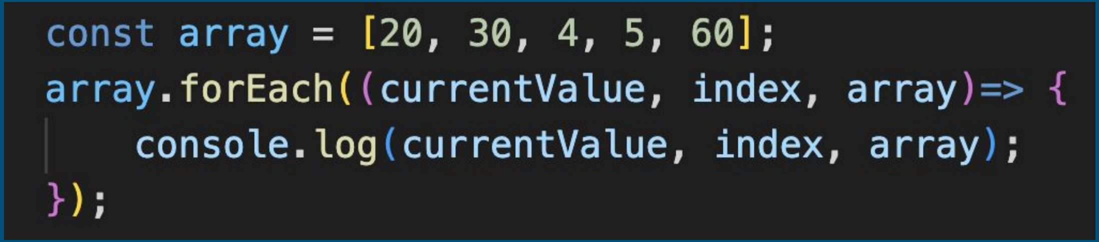
- Current Value (required) - The value of the current array element
- Index (optional) - The current element's index number
- Array (optional) - The array object to which the current element belongs

forEach( ) using callback



```
const array = [20, 30, 4, 5, 60];
array.forEach(function (currentValue, index, array) {
    console.log(currentValue, index, array);
});
```

forEach( ) using arrow function



```
const array = [20, 30, 4, 5, 60];
array.forEach((currentValue, index, array)=> {
    console.log(currentValue, index, array);
});
```

Assignment A: Using forEach( ) with arrow function and callback, File: 17_forEachAssigA.js

const arrayNumbers = [1, -7, 40, 502, -77, 91, 0, 108, 89, -601];

From the given arrayNumbers, try the hands-on for

1. Log the array element with it's index using forEach( ) with arrow function
2. Find the positive numbers and log on console
   a. Using forEach( ) with arrow function
3. Find the negative numbers, add into new array and and log new array on console using arrow function
4. Find the even numbers and log on console using forEach( ) with arrow function
5. Find the sum of all elements from arrayNumbers and log on sum value on console.
6. Log the only even indexed array value on console. forEach( ) with arrow function prefered

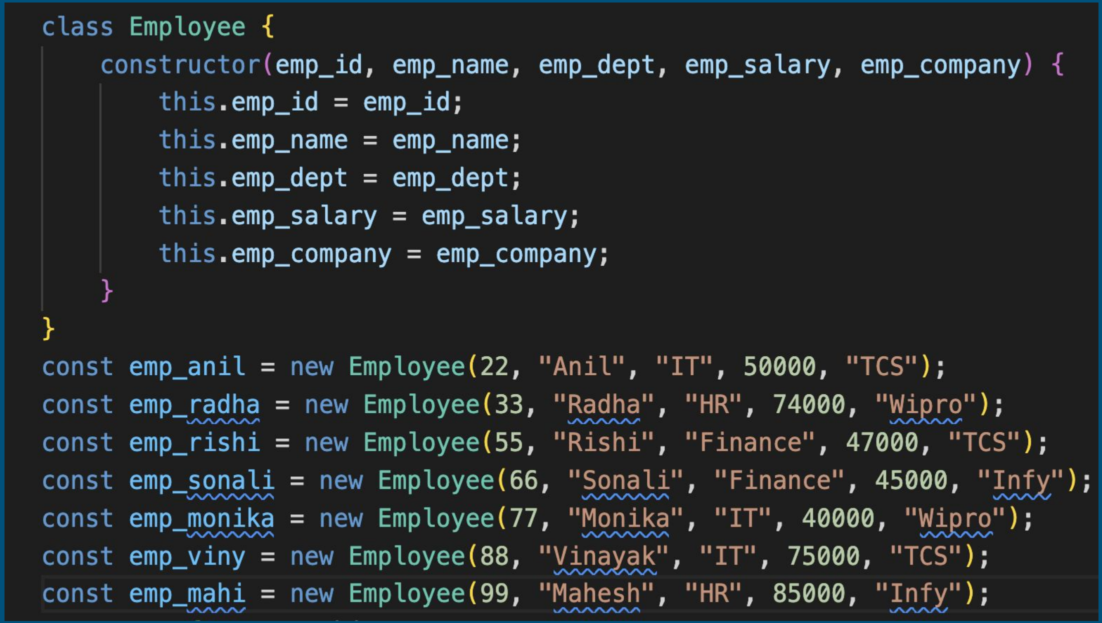
Assignment 0A: Use only forEach( )

File: 17_forEachAssigB.js

For the given employee objects

1. Find out the ‘TCS’ employee details and log only name & company on console

2. Find the employees with salary greater than or equal 50000 (Log the all employee details on console)



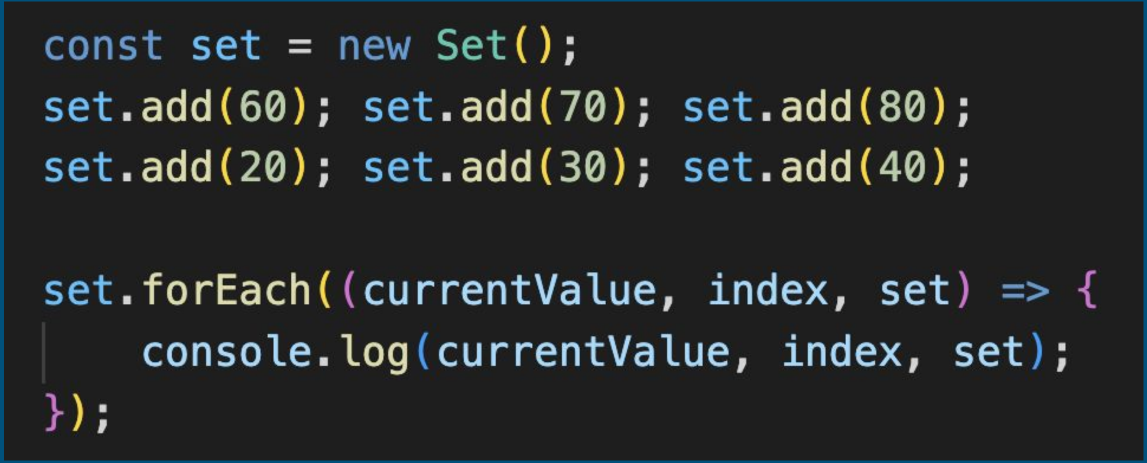
```
class Employee {
    constructor(emp_id, emp_name, emp_dept, emp_salary, emp_company) {
        this.emp_id = emp_id;
        this.emp_name = emp_name;
        this.emp_dept = emp_dept;
        this.emp_salary = emp_salary;
        this.emp_company = emp_company;
    }
}
const emp_anil = new Employee(22, "Anil", "IT", 50000, "TCS");
const emp_radha = new Employee(33, "Radha", "HR", 74000, "Wipro");
const emp_rishi = new Employee(55, "Rishi", "Finance", 47000, "TCS");
const emp_sonali = new Employee(66, "Sonali", "Finance", 45000, "Infy");
const emp_monika = new Employee(77, "Monika", "IT", 40000, "Wipro");
const emp_viny = new Employee(88, "Vinayak", "IT", 75000, "TCS");
const emp_mahi = new Employee(99, "Mahesh", "HR", 85000, "Infy");
```

3. Find the sum of all employees salary and log on console

4. Find the average salary and log on console (Average = Total Salary / Number of employees)

5. Find the ‘IT’ OR ‘HR’ department employees whose salary is greater than or equal to 75000 and log complete employee details on console

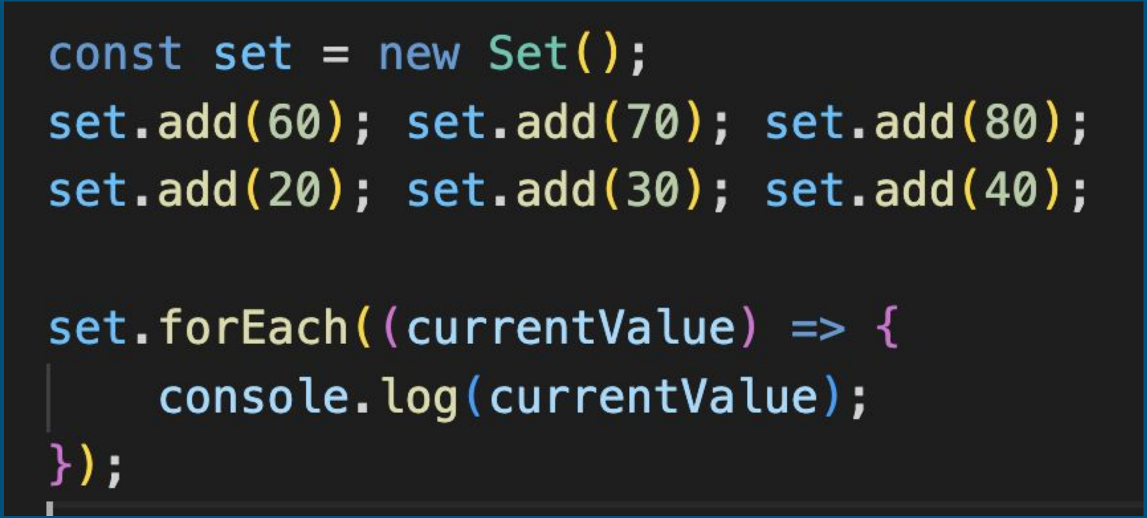
forEach( ) for Set



```
const set = new Set();
set.add(60); set.add(70); set.add(80);
set.add(20); set.add(30); set.add(40);

set.forEach((currentValue, index, set) => {
    console.log(currentValue, index, set);
});
```

As we know currentValue parameter is Only mandatory and other two are optional

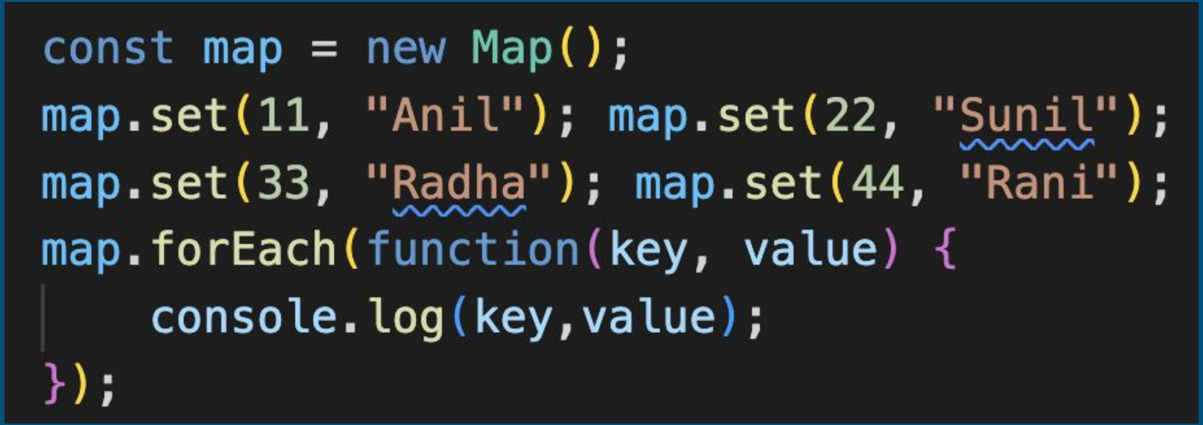


```
const set = new Set();
set.add(60); set.add(70); set.add(80);
set.add(20); set.add(30); set.add(40);

set.forEach((currentValue) => {
    console.log(currentValue);
});
```

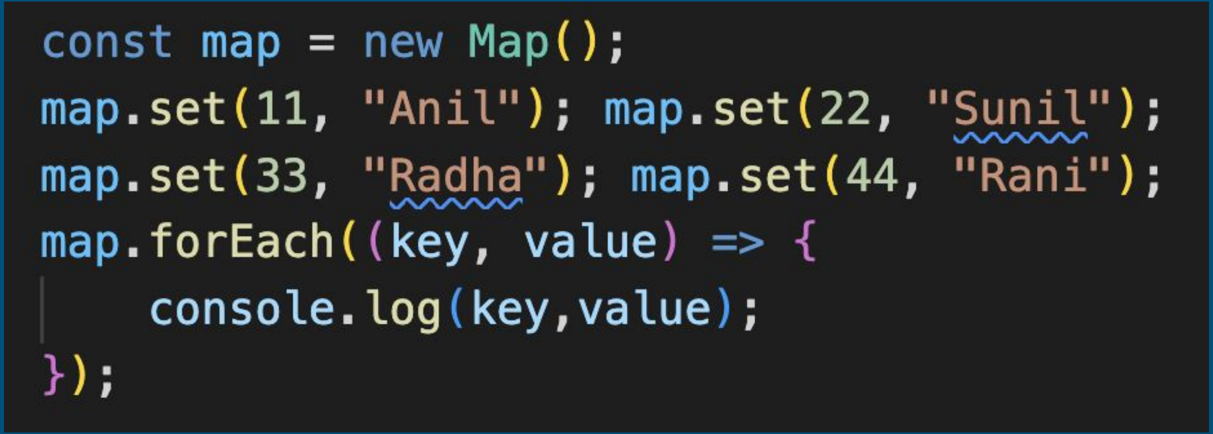
forEach( ) for map collection

forEach( ) for map using callback



```
const map = new Map();
map.set(11, "Anil"); map.set(22, "Sunil");
map.set(33, "Radha"); map.set(44, "Rani");
map.forEach(function(key, value) {
    console.log(key,value);
});
```

forEach( ) for map using arrow function



```
const map = new Map();
map.set(11, "Anil"); map.set(22, "Sunil");
map.set(33, "Radha"); map.set(44, "Rani");
map.forEach((key, value) => {
    console.log(key,value);
});
```

Assignment 0B: Create new file pls

File→ 15_ForEachMapAssigB.js

For the given Employee class, objects are created as shown in snippet

Create a Map Collection with name 'mapEmployees' and entries in such a way that employee id is the key and value is the employee object

Ex. 'mapEmployees'.set(22, emp_anil);

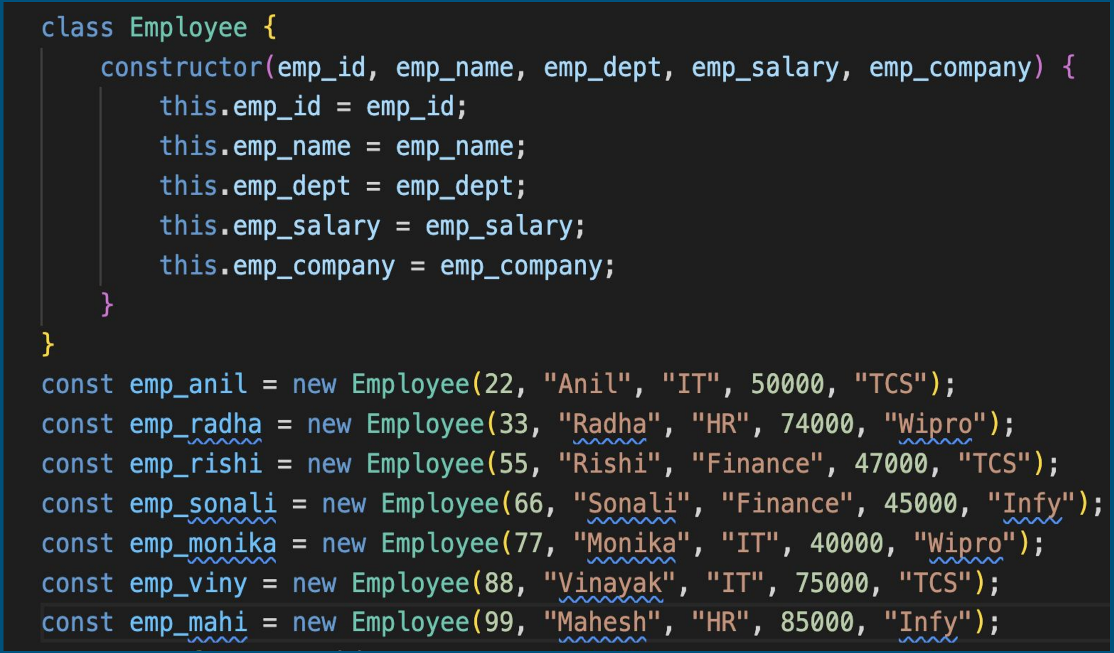
In this way add all entries

Log the details in format
Emp id ===> Name: emp_name , Dept: emp_dept, Company: emp_company, Salary: emp_salary

Please traverse using forEach() loop

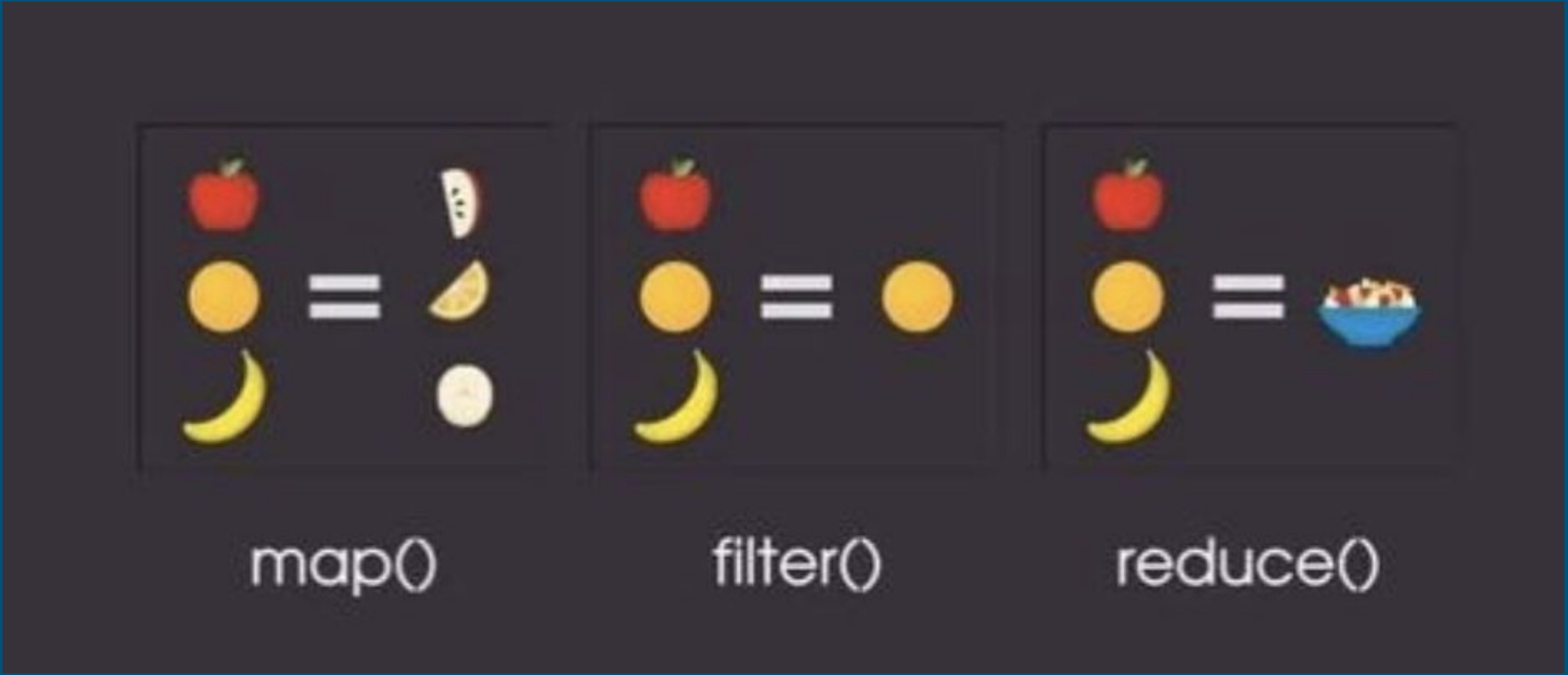
Example:

22 ===> Name: Anil, Dept: IT, Company: TCS, Salary: 50000

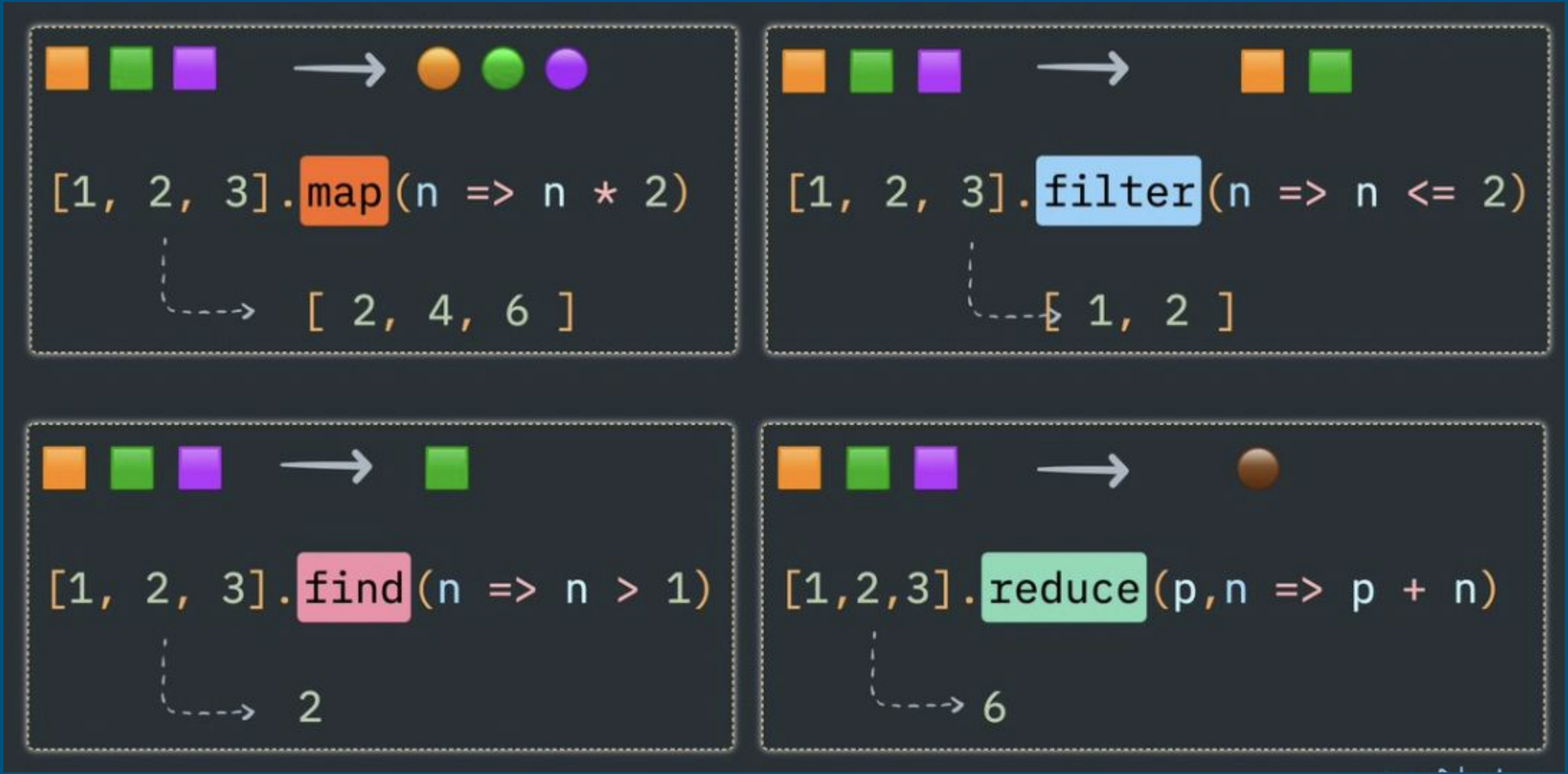


```
class Employee {
    constructor(emp_id, emp_name, emp_dept, emp_salary, emp_company) {
        this.emp_id = emp_id;
        this.emp_name = emp_name;
        this.emp_dept = emp_dept;
        this.emp_salary = emp_salary;
        this.emp_company = emp_company;
    }
}
const emp_anil = new Employee(22, "Anil", "IT", 50000, "TCS");
const emp_radha = new Employee(33, "Radha", "HR", 74000, "Wipro");
const emp_rishi = new Employee(55, "Rishi", "Finance", 47000, "TCS");
const emp_sonali = new Employee(66, "Sonali", "Finance", 45000, "Infy");
const emp_monika = new Employee(77, "Monika", "IT", 40000, "Wipro");
const emp_viny = new Employee(88, "Vinayak", "IT", 75000, "TCS");
const emp_mahi = new Employee(99, "Mahesh", "HR", 85000, "Infy");
```

map( ), filter( ) and reduce( ) methods visualization



map( ), filter( ), find( ) and reduce( ) methods visualization



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

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

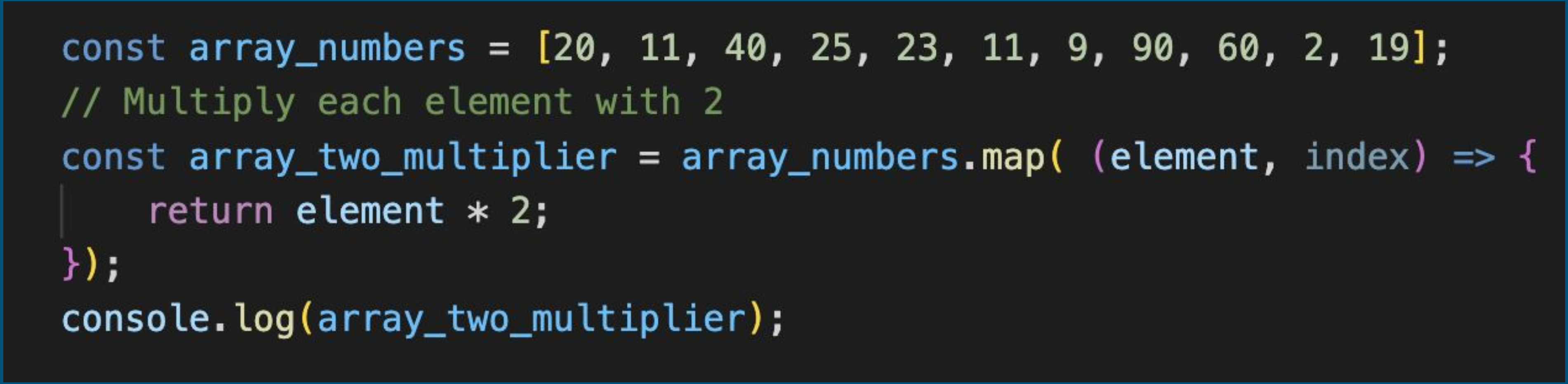
```
[1, 2, 3].find(n => n > 1)
  ---> 2
```

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

map( ) method: Used to transform the value. Allows to loop over array, access each value & returns the new array

map( ) method has three main arguments

- element / item (Mandatory) : Current array element the callback is iterating over
- index (Optional): Index of the current array element/item/value
- array (Optional): The array on which the map( ) method is being performed



```javascript
const array_numbers = [20, 11, 40, 25, 23, 11, 9, 90, 60, 2, 19];
// Multiply each element with 2
const array_two_multiplier = array_numbers.map( (element, index) => {
    return element * 2;
});
console.log(array_two_multiplier);
```

Assignment A: map( ): File→ 18_mapMethodAssigA.js

const arrayNumbers = [20, 11, 40, 25, 23, 11, 9, 31, 60, 2, 19];

For the given array Perform the below operations as

1. Add 10 into each element and log new array result on console
2. Square the each array element and log on console
3. Add the index value into its corresponding each array element and log new array result on console

AssignmentB: map( )

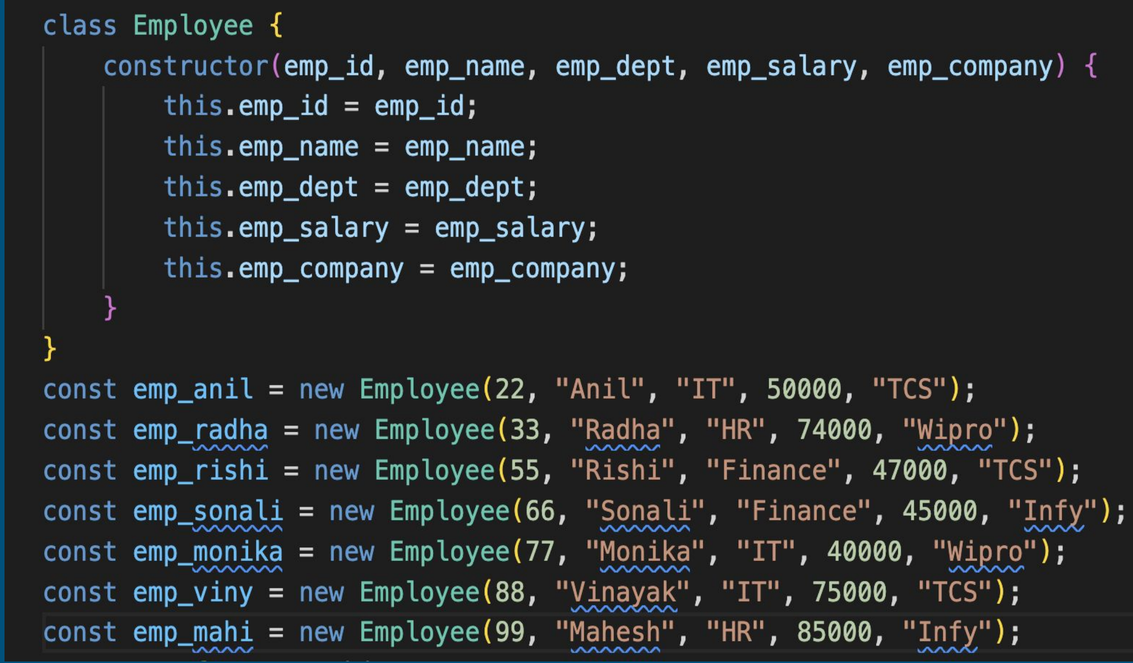
16_mapMethodAssigB.js

Add ~~all~~ employee objects inside array

1. Get the list of all employee names & log new array on console.

2. Get the list of departments & log on console

3. Get the list of employee id's and log on console



```
class Employee {
    constructor(emp_id, emp_name, emp_dept, emp_salary, emp_company) {
        this.emp_id = emp_id;
        this.emp_name = emp_name;
        this.emp_dept = emp_dept;
        this.emp_salary = emp_salary;
        this.emp_company = emp_company;
    }
}
const emp_anil = new Employee(22, "Anil", "IT", 50000, "TCS");
const emp_radha = new Employee(33, "Radha", "HR", 74000, "Wipro");
const emp_rishi = new Employee(55, "Rishi", "Finance", 47000, "TCS");
const emp_sonali = new Employee(66, "Sonali", "Finance", 45000, "Infy");
const emp_monika = new Employee(77, "Monika", "IT", 40000, "Wipro");
const emp_viny = new Employee(88, "Vinayak", "IT", 75000, "TCS");
const emp_mahi = new Employee(99, "Mahesh", "HR", 85000, "Infy");
```

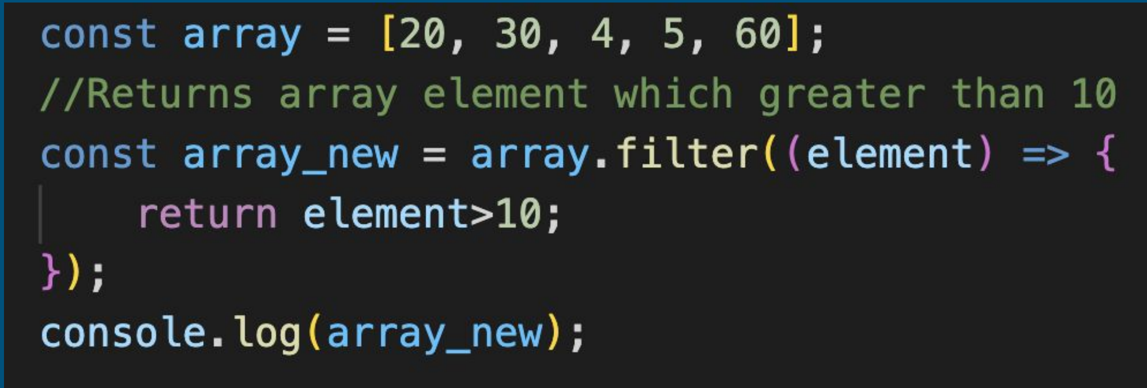
filter( ) method

Method filter( ) return a new array with all the elements that satisfies the condition that is passed inside the filter( ) method.

The array elements that does not satisfy the condition inside filter( ) are skipped and not included inside the new array

filter( ) method has three main arguments

- element / item (Mandatory) : Current array element the callback is iterating over
- index (Optional): Index of the current array element/item/value
- array (Optional): The array on which the filter( ) method is being performed



```
const array = [20, 30, 4, 5, 60];
//Returns array element which greater than 10
const array_new = array.filter((element) => {
    return element>10;
});
console.log(array_new);
```

Assignment B - for filter( ) method: File → 18_filterAssigA.js

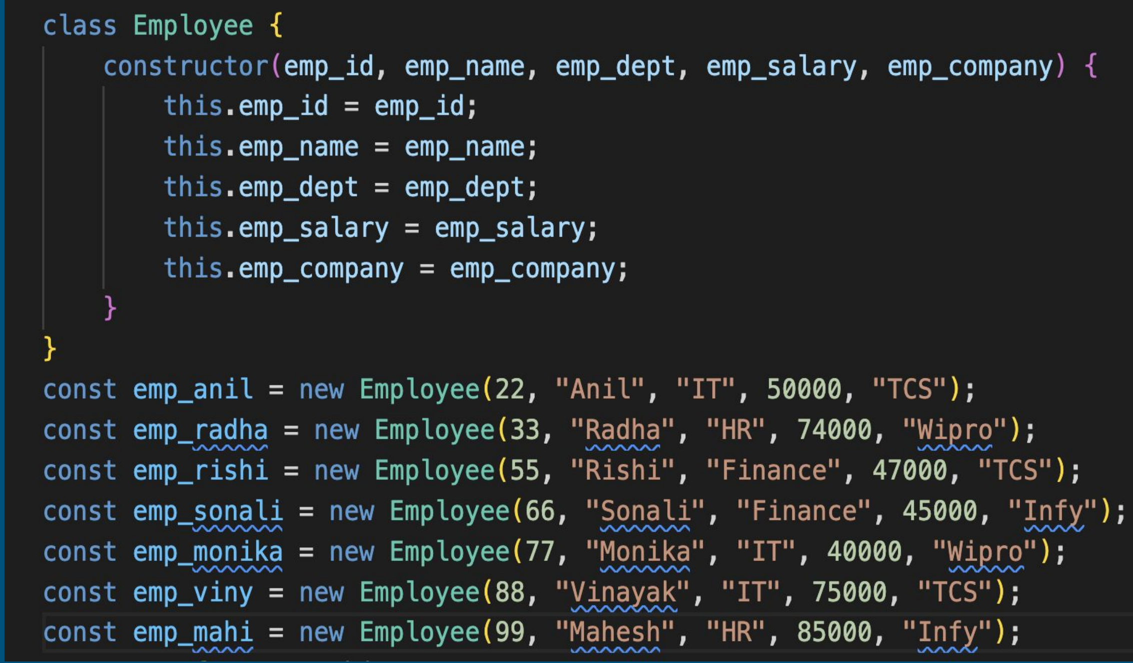
const arrayNumbers = [20, 11, 40, 25, 37, 49, 9, 90, 60, 2, 19];

1. Find out all the numbers which are greater than 50 and log on console
2. Find out all the even number and log on console
3. Find out all the odd numbers and log on console
4. Find out all the numbers which are multiple of 5
5. Find out all numbers which are between 20 and 50

Assignment: filter( )

Add the all employee objects inside array

1. Find out all the employees from 'TCS' using filter( ), and then from the Final array result log Company name & Employee name.

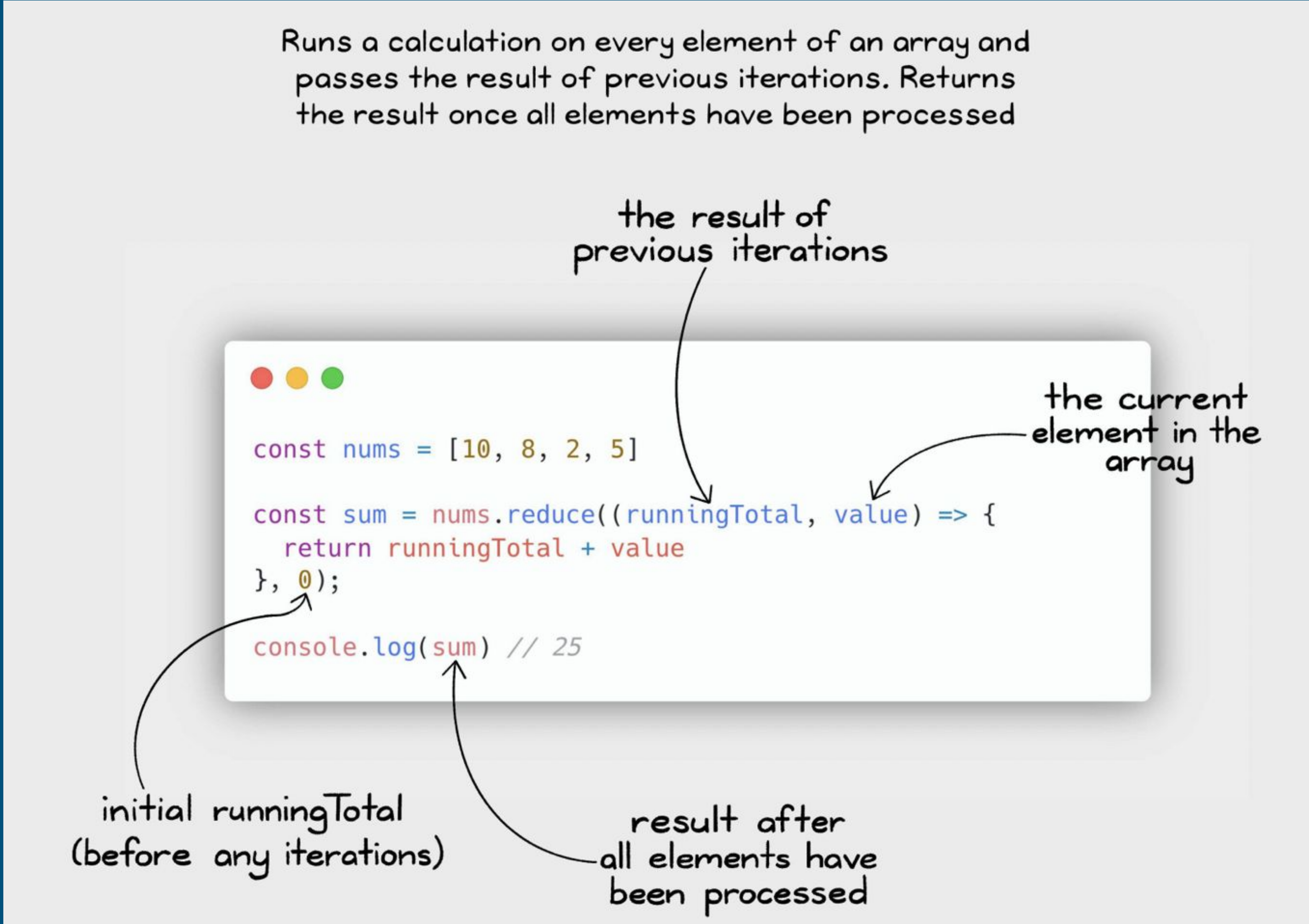


```
class Employee {
    constructor(emp_id, emp_name, emp_dept, emp_salary, emp_company) {
        this.emp_id = emp_id;
        this.emp_name = emp_name;
        this.emp_dept = emp_dept;
        this.emp_salary = emp_salary;
        this.emp_company = emp_company;
    }
}
const emp_anil = new Employee(22, "Anil", "IT", 50000, "TCS");
const emp_radha = new Employee(33, "Radha", "HR", 74000, "Wipro");
const emp_rishi = new Employee(55, "Rishi", "Finance", 47000, "TCS");
const emp_sonali = new Employee(66, "Sonali", "Finance", 45000, "Infy");
const emp_monika = new Employee(77, "Monika", "IT", 40000, "Wipro");
const emp_viny = new Employee(88, "Vinayak", "IT", 75000, "TCS");
const emp_mahi = new Employee(99, "Mahesh", "HR", 85000, "Infy");
```

2. Find the average salary of employee from company Wipro.

3. Find the average salary of employee from companies Wipro or Infy

reduce( ) method: Is used to reduce an array to a single value and iterate from left to right



Runs a calculation on every element of an array and passes the result of previous iterations. Returns the result once all elements have been processed

```
const nums = [10, 8, 2, 5]

const sum = nums.reduce((runningTotal, value) => {
  return runningTotal + value
}, 0);

console.log(sum) // 25
```

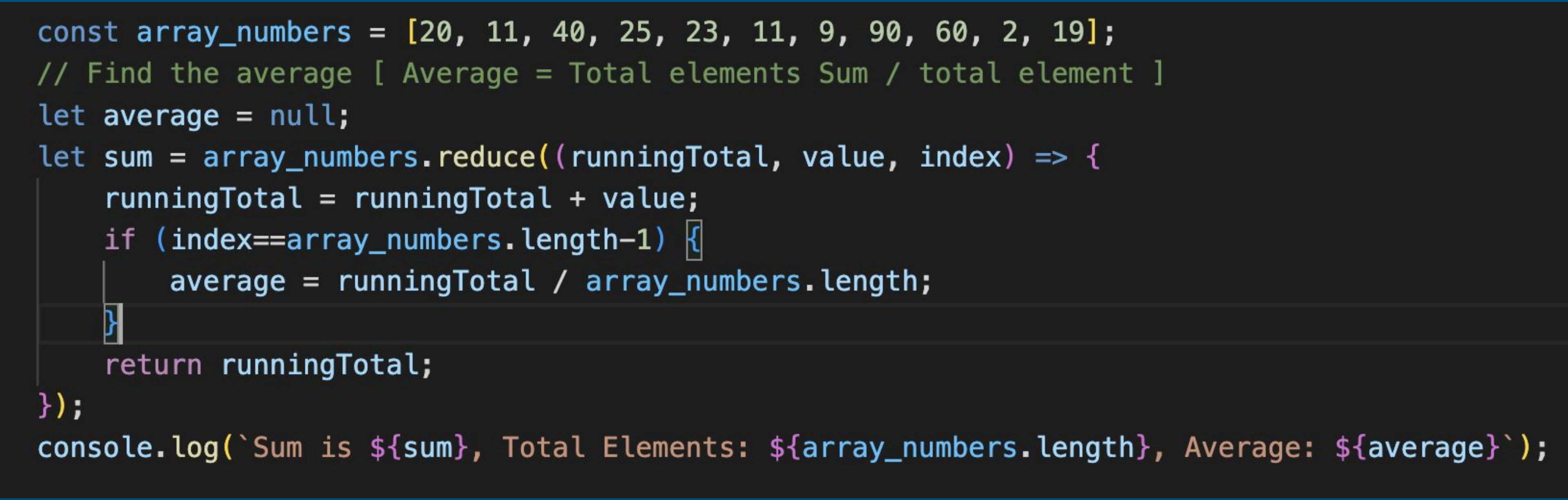
the result of previous iterations
the current element in the array
initial runningTotal (before any iterations)
result after all elements have been processed

Assignment C - for reduce( ) method

```
const array_numbers = [20, 11, 40, 25, 37, 49, 9, 90, 60, 2, 19];
```

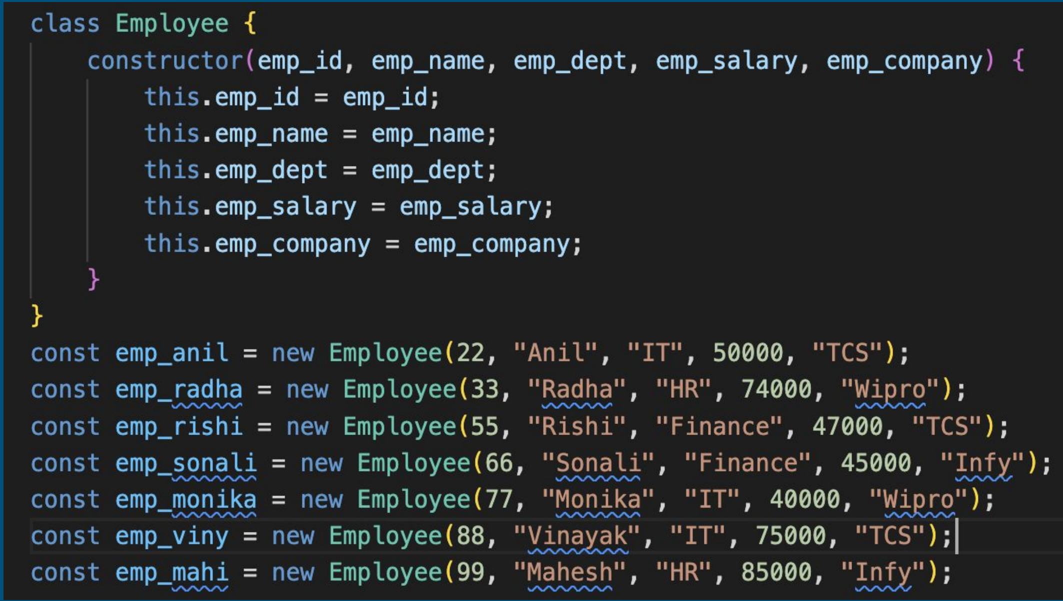
1. Find the sum of all numbers
2. Find the numbers multiple of 5 and then sum it[Hint → filter first then use reduce( )]

Find the average using reduce( )



```
const array_numbers = [20, 11, 40, 25, 23, 11, 9, 90, 60, 2, 19];
// Find the average [ Average = Total elements Sum / total element ]
let average = null;
let sum = array_numbers.reduce((runningTotal, value, index) => {
    runningTotal = runningTotal + value;
    if (index==array_numbers.length-1) {
        average = runningTotal / array_numbers.length;
    }
    return runningTotal;
});
console.log(`Sum is ${sum}, Total Elements: ${array_numbers.length}, Average: ${average}`);
```

Assignment D - Using filter( ) and reduce( )



```
class Employee {
    constructor(emp_id, emp_name, emp_dept, emp_salary, emp_company) {
        this.emp_id = emp_id;
        this.emp_name = emp_name;
        this.emp_dept = emp_dept;
        this.emp_salary = emp_salary;
        this.emp_company = emp_company;
    }
}
const emp_anil = new Employee(22, "Anil", "IT", 50000, "TCS");
const emp_radha = new Employee(33, "Radha", "HR", 74000, "Wipro");
const emp_rishi = new Employee(55, "Rishi", "Finance", 47000, "TCS");
const emp_sonali = new Employee(66, "Sonali", "Finance", 45000, "Infy");
const emp_monika = new Employee(77, "Monika", "IT", 40000, "Wipro");
const emp_viny = new Employee(88, "Vinayak", "IT", 75000, "TCS");
const emp_mahi = new Employee(99, "Mahesh", "HR", 85000, "Infy");
```

Given the Employee class

Add all employee object inside array namely 'arrayEmps'

1. Find all the employees from 'Wipro' company

2. Find all the employees from 'IT' OR 'HR' dept

3. Find all the employees whose emp id's are greater than 50

4. Find all the employees whose names start with letter 'A' or 'V' or 'M' [Hint→ startsWith("A") || startsWith("V") startsWith("M") inside if ( ) block as a condition]

5. Find the average salary of the employee for all the department

6. Find the average salary for 'IT' department [Hint → Filter the 'IT' department employee using filter( ) which return new array with only 'IT' department employees and then use reduce( ) to get the average]

Sorting the Array

Natural Sorting order

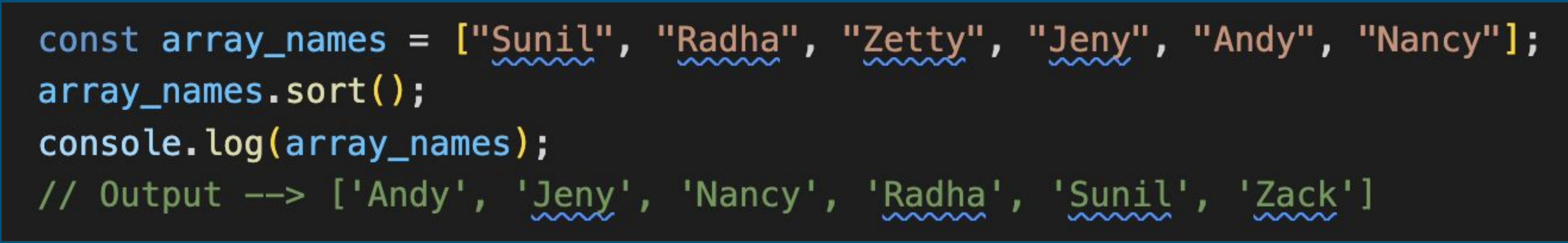
- For numbers → 1, 2, 3, 4 …… N (Ascending order)
  - N, (N-1), ….. 4, 3, 2, 1, 0 (Descending Order)
- For Alphabets → A, B, C ….. Z (Ascending order)
  - Z ….. , C, B, A (Descending Order)

Reverse

Insertion Order

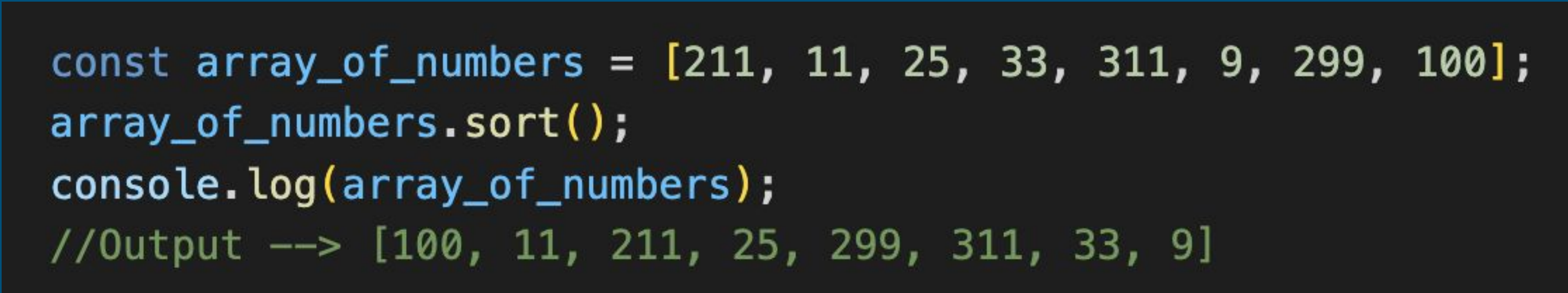
sort( ) method of array

Existing sort( ) method default nature for array of string values is, It sorts the names in ascending order, Which is expected and correct



```
const array_names = ["Sunil", "Radha", "Zetty", "Jeny", "Andy", "Nancy"];
array_names.sort();
console.log(array_names);
// Output --> ['Andy', 'Jeny', 'Nancy', 'Radha', 'Sunil', 'Zack']
```

while sorting an array of numbers when elements has more than two digits in that case sort( ) only considers first digit and hence the output is not as expected.



```
const array_of_numbers = [211, 11, 25, 33, 311, 9, 299, 100];
array_of_numbers.sort();
console.log(array_of_numbers);
//Output --> [100, 11, 211, 25, 299, 311, 33, 9]
```

sort( ) Working mechanism

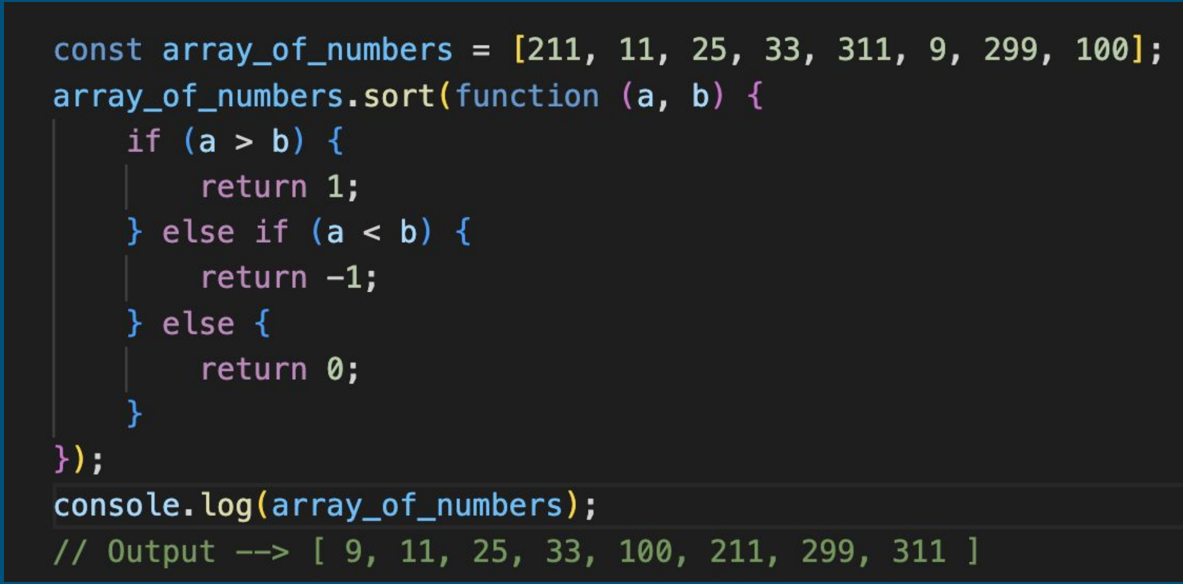
While comparing it accepts two arguments. Ex. a, b

If a > b :   Returns positive value

If a < b :   Returns negative value

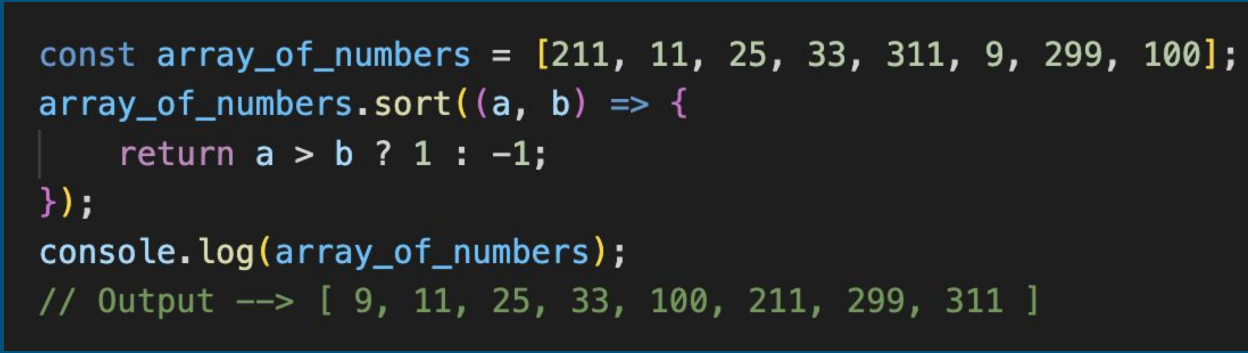
If a ==b : Returns zero

Sorting number array using *callback* function



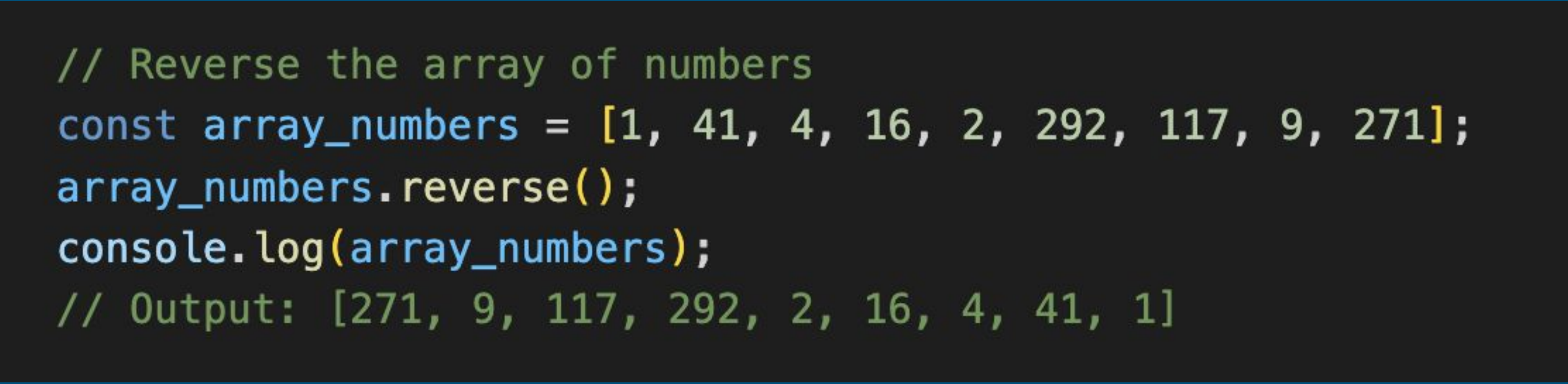
```
const array_of_numbers = [211, 11, 25, 33, 311, 9, 299, 100];
array_of_numbers.sort(function (a, b) {
    if (a > b) {
        return 1;
    } else if (a < b) {
        return -1;
    } else {
        return 0;
    }
});
console.log(array_of_numbers);
// Output --> [ 9, 11, 25, 33, 100, 211, 299, 311 ]
```

Sorting number array using *arrow* function

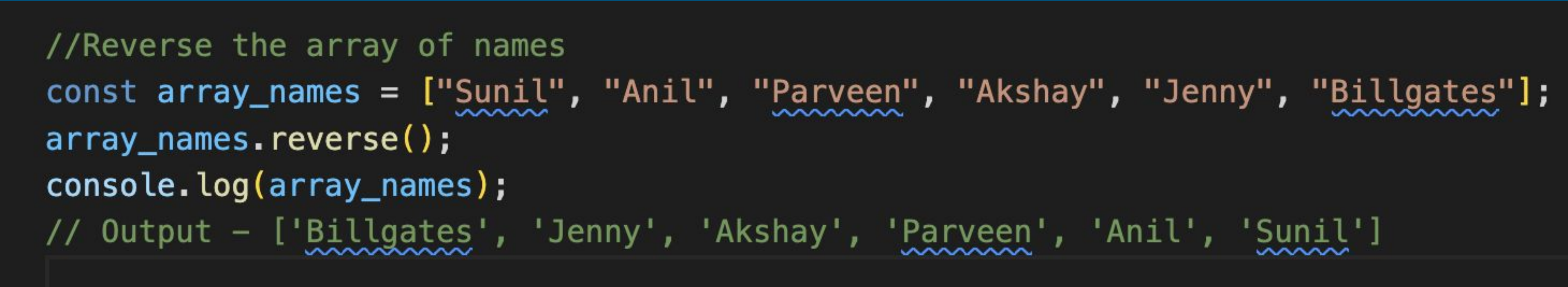


```
const array_of_numbers = [211, 11, 25, 33, 311, 9, 299, 100];
array_of_numbers.sort((a, b) => {
    return a > b ? 1 : -1;
});
console.log(array_of_numbers);
// Output --> [ 9, 11, 25, 33, 100, 211, 299, 311 ]
```

reverse( ): This method reverse the elements

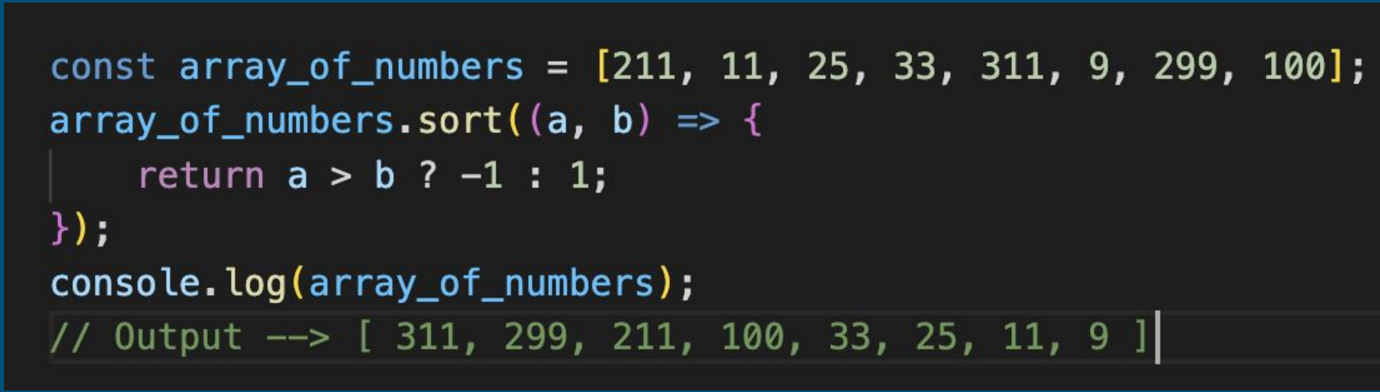


```javascript
// Reverse the array of numbers
const array_numbers = [1, 41, 4, 16, 2, 292, 117, 9, 271];
array_numbers.reverse();
console.log(array_numbers);
// Output: [271, 9, 117, 292, 2, 16, 4, 41, 1]
```



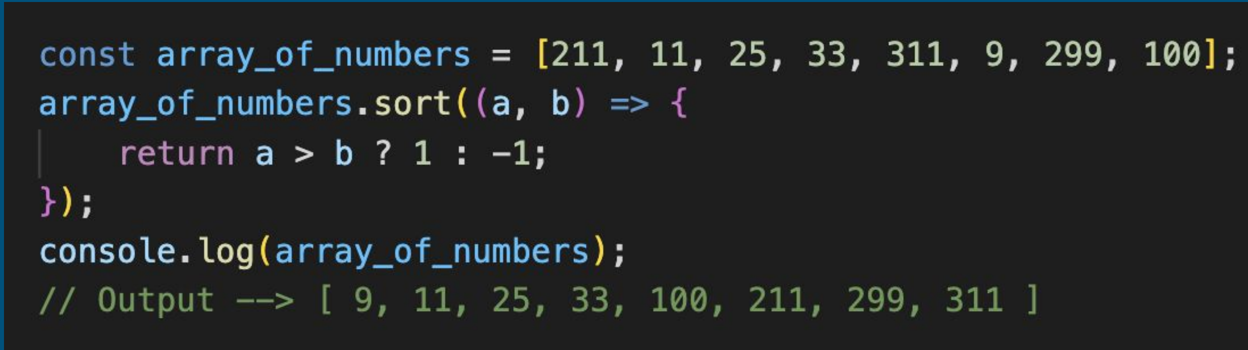
```javascript
//Reverse the array of names
const array_names = ["Sunil", "Anil", "Parveen", "Akshay", "Jenny", "Billgates"];
array_names.reverse();
console.log(array_names);
// Output - ['Billgates', 'Jenny', 'Akshay', 'Parveen', 'Anil', 'Sunil']
```

Sort the number array in descending order



```
const array_of_numbers = [211, 11, 25, 33, 311, 9, 299, 100];
array_of_numbers.sort((a, b) => {
    return a > b ? -1 : 1;
});
console.log(array_of_numbers);
// Output --> [ 311, 299, 211, 100, 33, 25, 11, 9 ]
```

Sorting number array using arrow function



```
const array_of_numbers = [211, 11, 25, 33, 311, 9, 299, 100];
array_of_numbers.sort((a, b) => {
    return a > b ? 1 : -1;
});
console.log(array_of_numbers);
// Output --> [ 9, 11, 25, 33, 100, 211, 299, 311 ]
```

Assignment A: Sorting in Ascending / Descending order and reverse

const arrayRollNumbers = [113, 45, 56, 11, 32, 45, 109, 799, 56, 45]

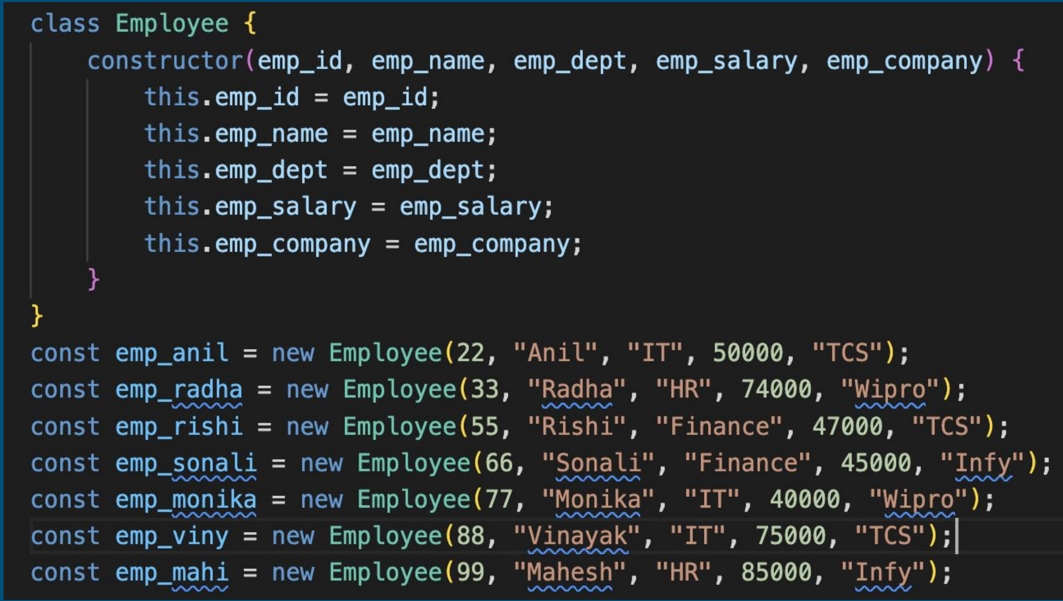
1. Reverse the array
2. Use the sort() method as it is without any custom sorting logic (Without passing any arguments) & notice the issue
3. Sort the array in ascending order, by writing your custom logic
4. Find the Greatest number from the array
5. Find the smallest number from the array
6. Remove duplicates from array

Note: After each step log the output on console

Assignment B: Sorting in ascending & Descending

For the given Employee class objects are Created please add all them in array. namely 'arrayEmployees'
Try the below:

1. Sort the 'arrayEmployees' in ascending order of Employee Id's and log employee details → Id, Name, Department



```
class Employee {
    constructor(emp_id, emp_name, emp_dept, emp_salary, emp_company) {
        this.emp_id = emp_id;
        this.emp_name = emp_name;
        this.emp_dept = emp_dept;
        this.emp_salary = emp_salary;
        this.emp_company = emp_company;
    }
}
const emp_anil = new Employee(22, "Anil", "IT", 50000, "TCS");
const emp_radha = new Employee(33, "Radha", "HR", 74000, "Wipro");
const emp_rishi = new Employee(55, "Rishi", "Finance", 47000, "TCS");
const emp_sonali = new Employee(66, "Sonali", "Finance", 45000, "Infy");
const emp_monika = new Employee(77, "Monika", "IT", 40000, "Wipro");
const emp_viny = new Employee(88, "Vinayak", "IT", 75000, "TCS");
const emp_mahi = new Employee(99, "Mahesh", "HR", 85000, "Infy");
```

2. Sort the 'arrayEmployees' in ascending order of employee department & log Id, dept, & Company

3. Sort the employee array in <u>descending</u> order of employee salary and log Name, Salary & Company

Thank you