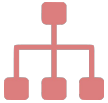
Data Structures: Objects

Relevel
by Unacademy

Concepts

What is an Array

Advantage and
Disadvantage of Array



Types of Arrays

Multidimensional Array

Different ways to
create an Array

Concepts

What is an Object

How to create an object

How to assign and access values of an object

Different ways to loop through an object

What is JSON

Concepts

Properties of JSON objects.

Application of JSON

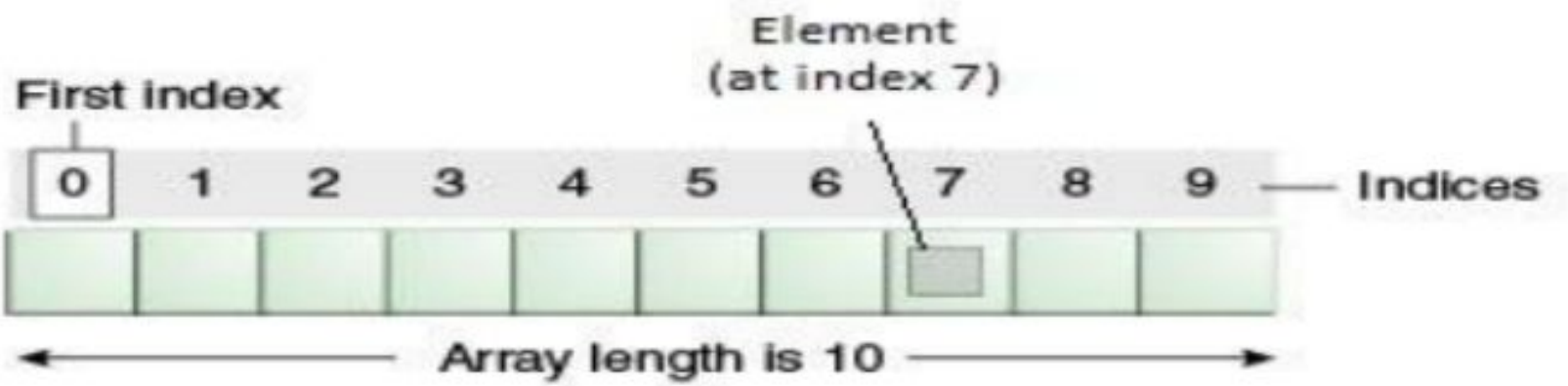
Difference in JSON Object and JavaScript Object.

Exercises

Arrays

- What is an array?
    - An array can be described as a group or collection of items (numbers) stored inside memory in a
    - logical order. The main function of Array is to store a collection of homogenous data of the same type.
    - In an array, all items are kept in a single memory region.
    - All arrays are allocated dynamically; it provides a variable-size list data structure, allowing elements to be removed or added in an array.
    - An array is a data structure where we save comparable objects.
    - An array is index-based, with the first element, or the first element in the Array, stored at the Oth index, the second element at the 1st index, the third element at the 2nd index, and so on.
    - An array can be used to store primitive values or objects in JavaScript.
    - We can create both single-dimensional and multidimensional arrays.

The following is the representation of an array:



What are the Advantage(s) and Disadvantage(s) of Array?

Advantage

- Arrays represent numerous data items of a similar type using a single name.
- In arrays, the elements can be accessed in any order by using the index number.
- Arrays allocate memory in contiguous memory locations for every element present in an array. Henceforth there is no possibility of additional memory being allocated in the case of arrays. This saves memory spillage or overflow of memory and helps to regulate memory usage in any database.
- With the use of arrays, other data structures like stacks, queues, trees, linked lists, graphs, etc., can be implemented.
- Arrays represent numerous data items of a similar type using a single name.

What are the Advantage(s) and Disadvantage(s) of Array?

Disadvantage

- Size Limit: In the Array, we can only store elements of a fixed size. It does not expand in size during use.
- In JavaScript, a collection framework is employed to handle this problem, which grows automatically.

Example 1 (Without Array)

- In the below example, we will repeat the same code with different values without using an array.

```
// Here, we are creating five variables to store five different cities
const city1 = 'London';
const city2 = 'Mumbai';
const city3 = "Chennai';
const city4 = 'Bangalore'; %3!
const city5 = 'Kerala';

// To print all the elements to the console
console.log(city1); // London
console.log(city2); // Mumbai
console.log(city3); // Chennai
console.log(city4); // Bangalore
console.log(city5); // Kerala
```

Example 2 (Without Array)

- Here in this example, we will run the same code using an Array.

```
// To create an array of cities to store values
const cities = ['London', 'Mumbai', 'Chennai', 'Bangalore', 'Kerala']
// To print all the elements entered in a array to the console
for (let i = 0; i < cities.length; i++) {
console.log (cities [i]);
}
London
Mumbai
Chennai
Bangalore
Kerala
```

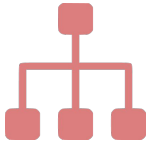
The main difference between with Array and Without Array are,

1. Without array each value are stored in different variables which result in different memory location, whereas using array the values are stored in contiguous memory allocation
2. For developer we can easily maintain code and data manipulation is good for larger data, without array we need to create 100 variables for 100 values but in array we can insert the values in the single array

Note:

1. When we have to create a database with complex and multiple variables to store, it will be difficult to control a database without using an array.
2. Using Arrays in the database makes it easier to manage and control the database in the programming language; we can easily create an array and enter or remove values in it.

What are the types of Array?



Arrays can be divided into two types.

Array with a single dimensional or one-dimensional array

A multidimensional Array (MDA)

Array with a single dimension

We can declare and allocate memory for an array in a single statement in JavaScript.

Here is an **example**,

```
// To create an Single-Dimension array:
var colors = [" Blue "," Green "," Yellow "];
// To print the elements in the console:
console.log(colors);
["red", "blue", "green"]
```

The syntax for declaring an single dimension array is as follows:

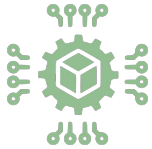
```
var <name_of_array> = [element_0, element_1, element_2, element_3, ...
elementN];
```

Different approach to create array in JavaScript:

```
// To create an Single-Dimension array:
const myArray = [ 'h', 'e', 'l', 'l' , 'o' );
// first element
console.log(myArray[0]); // to print "h"
// second element
console.log(myArray[1]); // to print "e"
// third element
console.log(myArray[2]); // to print "l"
// fourth element

console.log(myArray[3]); // to print "l"
```

```
// fifth element
console.log (myArray[4]); // to print "o"
Output:
"h"
"e"
"l"
"l"
"o"
```

What are Multidimensional Arrays ?



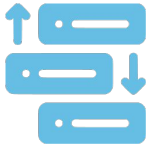
- A Multidimensional Array can be described as an array that consists of two or more than two-dimensional Array.
- The arrays we've discussed so far are known as one-dimensional arrays. In JavaScript, however, multidimensional arrays can also be declared.
- A multidimensional array is also termed as Array of arrays.
- To build a two-dimensional array, wrap each Array in its own pair of " [] "square brackets.
- In other words, each element of a multidimensional array is an array in and of itself.

Example for Multi_Dimensional Array :

```
//example to create multi-dimensional Array
var items = [
    [2, 31],
    [5, 6],
    [7, 8]
];
console.log(items[0][0]); // 1
console.log(items[0][1]); // 2
console.log(items[1][0]); // 3
console.log(items[1][1]); // 4
console.log(items);
```

```
Output:
2,
3,
5,
6
[[2, 31, [5, 6], [7, 81]
```

What are different ways to create Arrays in JavaScript?



In Javascript, there are many ways in which an array can be created

- **Array Literals -**

The easiest and the most common way of creating an array is using Array Literals.

Example:

```
const cities = ["London"," Mumbai"," Orange"," Cherry"]
```

- **New Keyword**

Using the new Javascript keyword, we can create a new array.
Example:

```
const cities = new Array("London"," Mumbai"," Cherry"," Kerala ")
```

Common Operations performed on Arrays:

Looping through Array:

There are many ways to loop over the Array. The most common ways of looping through arrays in Javascript are:

For Loop :

We saw a previous example above. Let's see one more and get familiar with the syntax.

```
// Creating an Array
const cities = ['Mumbai', 'Delhi', 'Kolkata', 'Lucknow', 'Chennai'];
// Using loop to iterate over array
for (let i = 0; i < cities.length; i++)
// Printing array elements using index
  console.log (cities [i]);
```

```
// Mumbai
// Delhi
// Kolkata
// Lucknow
// Chennai
```

Length of array:

```
const cities = ["London", "Mumbai"," Bangalore"," Kerala"]
let length =cities.length
console.log (length) 4
```

Note: Array index always starts from 0, which means the first element is stored at index 0 and the last element will be stored at index length-1.

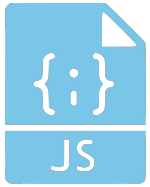
Objects in Real World

- In the real world any non-living entity can be referred to as an **Object**.
- A car,bike, smartphone, laptop, chair, table and any simplest thing you can think of is basically an object.
- These objects have some property and functionality. Consider a smartphone.
- Its properties
    - Colour
    - Size of the screen
    - Storage
    - Camera
    - Battery

Objects in Real World

- Its functionalities
    - Calling
    - Running applications
    - Browsing
    - Taking pictures/videos
- In the same way objects are used in Javascript or in programming in general to represent it as a real world object. These objects also possess some properties and functionalities more commonly referred to as methods/functions which are discussed below.

Objects in Js



- Object in Javascript is an entity which has properties and methods associated with it. These objects can have properties in the form of “key : value” pairs, here key is a string which can also be referred to as property name and value can be anything (e.g. string, number, null, array, function etc.).

- **Everything in Javascript is an object except the primitive data types that include number, string, boolean, null, undefined.**

```
// Creating object using Object literal

const laptop = {
  make: 'Dell',
  model: 'Alienware',
  memory: ['SSD', 'HDD'],
  cores: 8,
  memorySize: [256, 512],
};
```

Creating Objects using Object literal

1. The easiest and the most common way of creating an object is using Object Literals

```
// Creating object using Object literal

const laptop = {
  make: 'Dell',
  model: 'Alienware',
  memory: ['SSD', 'HDD'],
  cores: 8,
  memorySize: [256, 512],
};
```

Creating Objects using New Keyword

2. Using the new Javascript keyword, we can create a new object.

```
// Creating object using new keyword

const laptop = new laptop();
laptop.make = 'Apple';
laptop.model =  'MacBook Pro';
laptop.cores = 8;
```

Creating Objects using New Keyword

2. Using the new Javascript keyword, we can create a new object.

```
// Creating object using new keyword

const laptop = new laptop();
laptop.make = 'Apple';
laptop.model =  'MacBook Pro';
laptop.cores = 8;
```

Creating Objects using Constructor and this keyword

Using this keyword and object constructor, we can create a new object.

```
// Creating object using object constructor and this keyword

// Laptop constructor
function Laptop(make, model, cores) {
  this.make = make;
  this.model = model;
  this.cores = cores;
}

// Creating the object by calling the constructor
const myLaptop = new Laptop('Apple', 'MacBook Pro', 8);
```

Object.create method

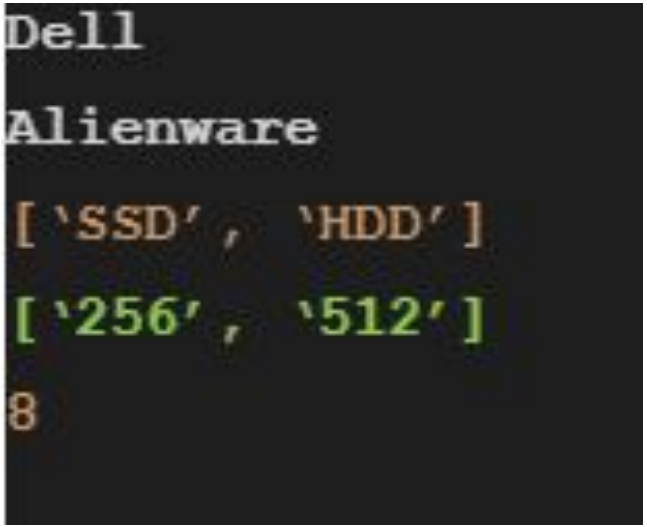
There is an inbuilt function for creating an object in javascript
Example:

```
// Creating object using create method

const laptop = {
  make: 'Dell',
  model: 'Alienware',
  memory: ['SSD', 'HDD'],
  cores: 8,
  memorySize: [256, 512],
};

const laptopObj = Object.create(laptop);
```

Accessing properties of objects in JS

```
const laptop = {
  make: 'Dell',
  model: 'Alienware',
  memory: ['SSD', 'HDD'],
  cores: 8,
  memorySize: [256, 512],
};


console.log(laptop.make);
console.log(laptop.model);
console.log(laptop.memory);
console.log(laptop.memorySize);
console.log(laptop.cores);
```

Output



Looping through an object

Using a for...in a loop

The most common and easy way to implement loop through an object's properties is by using the **for...in statement :**
Here is an example that implements for...in loop over an object:

```
const array_for_userdata = {
    name: 'Ben Accord',
    email: 'ben.english@example.com',
    age: 25,
    dob: '08/12/1996',
    active: true
};
// iterate over the user object

for (const key in user)
{
```

```
        console.log(`${key}:                ${array_for_userdata[key]}`);
}

// name: John Doe

// email: ben.english@example.com

// age: 25

// dob: 08/12/1995

// active: true
```

```
// [ 'java', 'javascript', 'nodejs', 'php' ]

// iterate over object

keys.forEach((key, index) => {

    console.log(`${key}: ${array_name_for_courses[key]}`);

});

Output:

// java: 15

// javascript: 78

// nodejs: 38

// php: 96
```

Object.entries () method

This the third method konwn as Object.entries() another method that can be used for traversing an array. .
Object.entries() gives outputs of an array of arrays that consists of each inner array having two elements. The first element is considered being the property and the second element is the value.

```
const array_animals = {
lion: 1,
giraffe: 2,
tiger: 3,
elephant: 4
};

const entries = Object.entries(array_animals);
console.log(entries);
```

```
Output:

//             [             [             'lion',             1             ],
//                             [             'giraffe',             2             ],
//                               [             'tiger',             3             ],
//    [ 'elephant', 4 ] ]
```

Object.values () method

Looping through The Object.values() method works directly opposite to that of Object.key(). Object.values() method functions by returning the values of all properties in the object as an array. We can also then loop through the values of the array by using any of the array looping methods.

Example for Object.values() method:

```
const array_of_animals = {
lion: 1,
horse: 2,
giraffe: 3,
elephant: 4
};
// iterate over object values
Object.values(array_of_animals).forEach(val => console.log(val));
```

```
Output:

//                                                                  1
//                                                                  2
//                                                                  3
//  4
```

Object.assign method

Object.assign method is used to assign one or more source object and to form a new object.

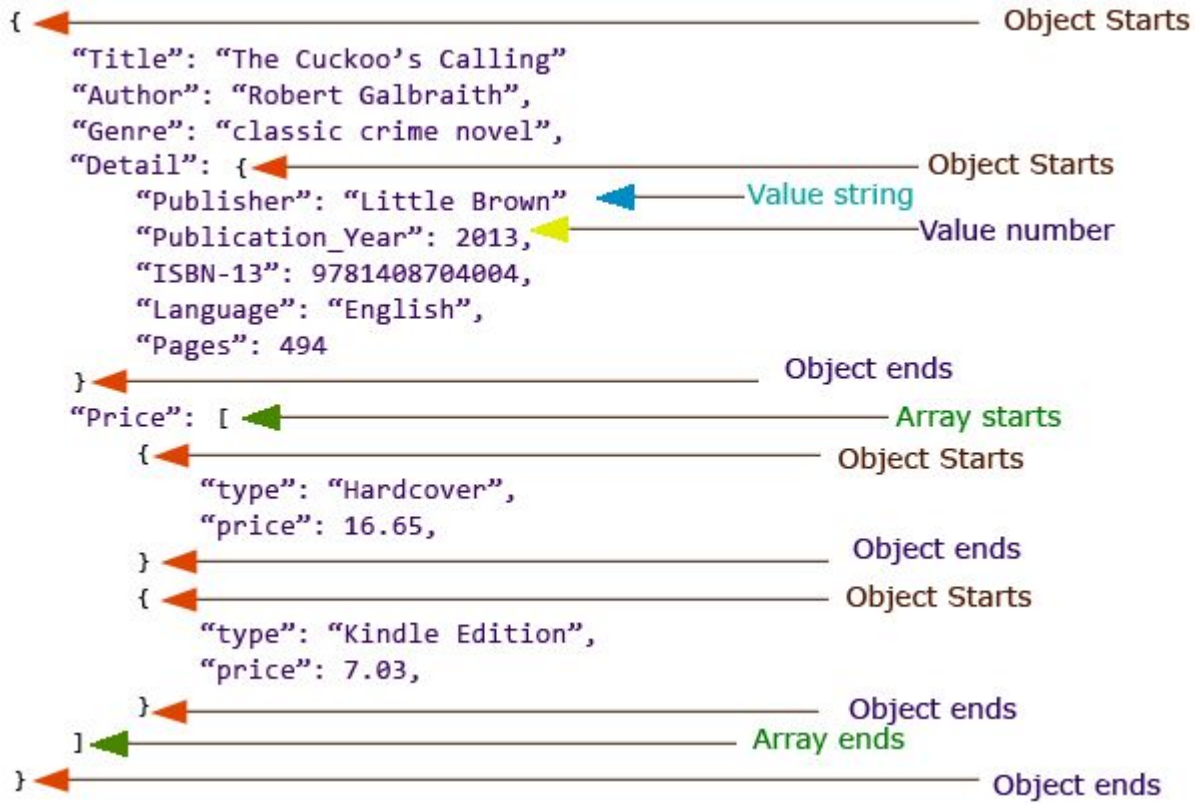
```
// clone object using assign method

const laptop = {
  make: 'Dell',
  model: 'Alienware',
  memory: ['SSD', 'HDD'],
  cores: 8,
  memorySize: [256, 512],
};

const laptopObjCopy = Object.assign({}, laptop);
```

Object.keys () method

The Object.keys() method makes it easier to loop over objects.

Object.keys () method takes the object that we want to to loop over as an argument and returns the elements in an array that contains all properties names (or keys).

```
const array_name_for_courses = {
   java: 15,
  javascript: 78,
nodejs: 38,
php: 96 };

// to implement conversion object to key's array

const keys = Object.keys(array_name_for_courses);

// to print all keys

console.log(keys);
```


{
"Title": "The Cuckoo's Calling"
"Author": "Robert Galbraith",
"Genre": "classic crime novel",
"Detail": {
"Publisher": "Little Brown"
"Publication_Year": 2013,
"ISBN-13": 9781408704004,
"Language": "English",
"Pages": 494
}
"Price": [
{
"type": "Hardcover",
"price": 16.65,
}
{
"type": "Kindle Edition",
"price": 7.03,
}
]
}
Object Starts
Object Starts
Value string
Value number
Object ends
Array starts
Object Starts
Object ends
Object Starts
Object ends
Array ends
Object ends

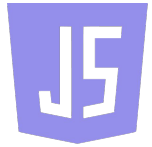
Accessing properties of JSON objects

The Object.keys() method makes it easier to loop over objects.

Object.keys () method takes the object that we want to to loop over as an argument and returns the elements in an array that contains all properties names (or keys).

```
const data = {
    "car": "Audi",
    "models": ["Q7","Q5"],
    "launchYear": 2021,
    "price": [5000000, 3500000]
};
console.log(data);
console.log(data.car);
console.log(data.models[0]);
```

Difference between JSON Object and Javascript Object:



- Though JSON Object and Javascript Object have a close resemblance as both are key:value pairs there are some things we need to note
- Property name / key are always in double quotes " "
- The keys are any valid string but the JSON values can only be one of the six data types (strings, numbers, objects, arrays, boolean, null). Unlike in Javascript Objects the values can literally be anything as we saw earlier.

Application of JSON

Here are some common applications of JSON:

- Helps you to transfer data from a server to client
- Sample JSON file format helps in transmit and serialize all types of structured data.
- Allows you to perform asynchronous data calls without the need to do a page refresh
- It is widely used for JavaScript-based application, which includes browser extension and websites.
- You can transmit data between the server and web application using JSON.
- We can use JSON with modern programming languages.
- It is used for writing JavaScript-based applications that include browser add-ons.
- Web services and Restful APIs use the JSON format to get public data.

Exercises

1. Javascript Code remove the duplicate element in the array

Input: `const arr = ['a', 'b', 'c', 'a', 'c', 'd', 'a'];`

Output: `[ 'a', 'b', 'c', 'd' ]`

Explanation:

1. Create a object to store the array values and their occurrence counts
2. Iterate the array and add the array value in the object as key and assign 0 as value if the key is not present in the object
3. If the array value is already present as key in the object then add the values with
4. Use the Object.keys() function to get the keys as an array

JS Code

Checkout to code link for reference.

```
const removeDuplicate = (arr) => {
    const obj = {};
    arr.map(item => {
        if (obj[item])
            obj[item] += 1;
        else
            obj[item] = 0;
    })
    return Object.keys(obj);
}

const arr = ['a', 'b', 'c', 'a', 'c', 'd', 'a'];
console.log(removeDuplicate(arr));
```

2. JavaScript practice program to iterate the object and log the object by framing the meaningful sentence

Explanation for the above program:

Here we have used an object called car and define the below property

a) Color – black
b) Speed – 120Kmph
c) Brand – Audi
d) Start and stop as function

calling the property and functionality by framing the meaning full sentence
Using Object.entries(object) to get the object individually from the array and iterate using for loop

Output:
My car is Audi and color is Black
My car is BMW and color is Red

Javascript Code:

Checkout to code link for reference.

```
const car = [{
    color: 'Black',
    speed: '120Kmph',
    brand: 'Audi',
    start: function () {
        console.log('Car started');
    },
    stop: function () {
        console.log('Car stopped');
    },
},
{
```

```
    color: 'Red',
    speed: '100Kmph',
    brand: 'BMW',
    start: function () {
        console.log('Car started');
    },
    stop: function () {
        console.log('Car stopped');
    },
},]

for ([key, value] of Object.entries(car)){
    console.log(`My car is ${value.brand} and color is ${value.color}`)
}
```

3. JavaScript practice program to get the number of times the particular letter is occurring in the given string

Input: OCCURRENCE

Output:

O occurring 1 times
C occurring 3 times
U occurring 1 times
R occurring 2 times
E occurring 2 times
N occurring 1 times

Explanation:

1. split the string using split() function and it will give the array of letters
2. declare an empty object
3. Iterate the array and add the letter in the object as key and assign 0 as value if the key is not present in the object
4. If the letter is already present as key in the object then add the values with 1
5. Finally iterate the object and log the occurrence

Code:

Checkout to code link for reference.

```
function getOccurrence (str) {
  const splitStr = str.split('');
  const occurrenceObj = {};
  splitStr.map(item => {
    if (occurrenceObj[item]) {
      occurrenceObj[item] += 1;
    }
    else{
      occurrenceObj[item] = 1;
```

```
    }
  })
  for (let [value, index] of Object.entries(occurrenceObj)){
    console.log(`${value} occurring ${occurrenceObj[value]} times`);
  }
}
getOccurrence("OCCURRENCE")
```

4. **Write a program to Count of Right-Angled Triangle formed from given N points whose base or perpendicular are parallel to X or Y axis**

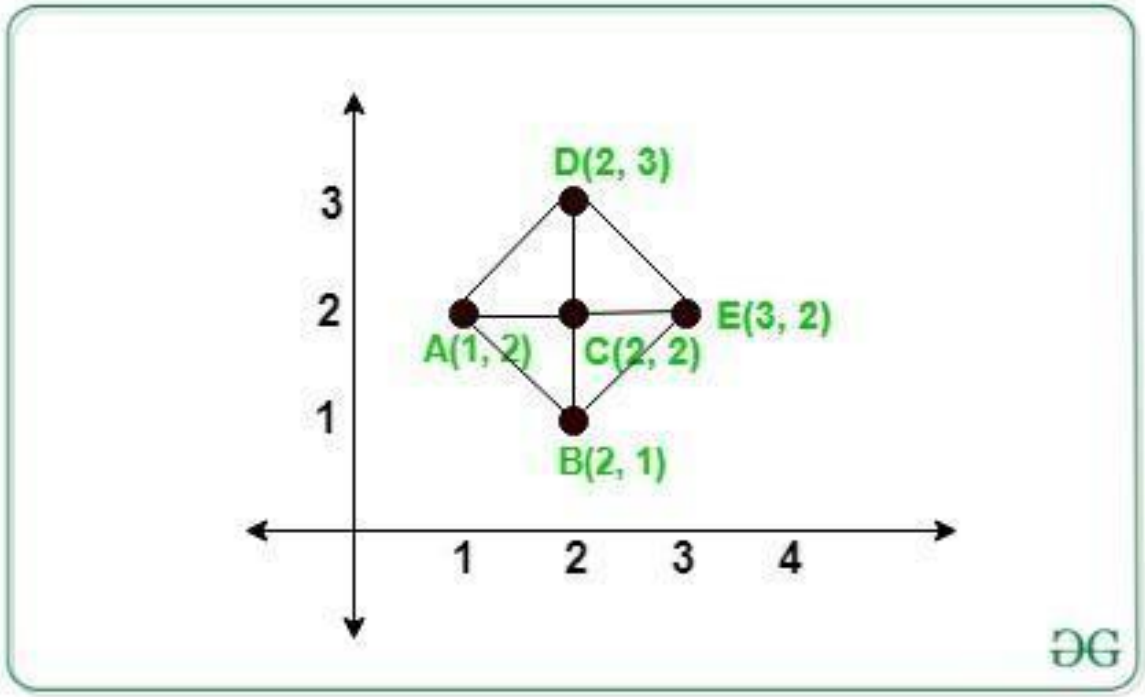
Input: arr[][] = {{1, 2}, {2, 1}, {2, 2}, {2, 3}, {3, 2}}

Output: 4

Explanation:

Given an array arr[] of N distinct integers points on the 2D Plane. The task is to count the number of Right Angled Triangle from N points such that the base or perpendicular is parallel to the X or Y-axis.

Count of right-angled triangles = (frequencies of X coordinates – 1) * (frequencies of **Y** coordinates – 1)


D(2, 3)
E(3, 2)
A(1, 2)
C(2, 2)
B(2, 1)
3
2
1
1
2
3
4

Below are the steps:

Create two maps to store the count of points, one for having the same **X-coordinate** and another for having the same **Y-coordinate**.

For each value in the map of **x-coordinate** and in the map of **y-coordinate** choose that pair of points as pivot elements and find the frequency of that pivot element.

For each pivot element(say **pivot**) in the above step, the count of right-angled is given by

```
(m1[pivot].second-1)*(m2[pivot].second-1)
```

Similarly, calculate the **total possible right-angled triangle** for other **N** points given.

Finally, **sum all the possible** triangle obtained that is the final answer.

Checkout to code link for reference.

```
// Function to find the number of right
// angled triangle that are formed from
// given N points whose perpendicular or
// base is parallel to X or Y axis

function RightAngled(a, n) {
    // To store the number of points
    // has same x or y coordinates

    var xpoints = {};
    var ypoints = {};
```

```
for (var i = 0; i < n; i++) {
  if (xpoints.hasOwnProperty(a[i][0])) {
    xpoints[a[i][0]] = xpoints[a[i][0]] + 1;
  } else {
    xpoints[a[i][0]] = 1;
  }

  if (ypoints.hasOwnProperty(a[i][1])) {
    ypoints[a[i][1]] = ypoints[a[i][1]] + 1;
  } else {
    ypoints[a[i][1]] = 1;
  }
```

```
}

// Store the total count of triangle
var count = 0;

// Iterate to check for total number
// of possible triangle

for (var i = 0; i < n; i++) {
  if (xpoints[a[i][0]] >= 1 && ypoints[a[i][1]] >= 1) {
    // Add the count of triangles
    // formed
    count += (xpoints[a[i][0]] - 1) * (ypoints[a[i][1]] - 1);
  }
}
```

```
  // Total possible triangle
  return count;
}

// Driver Code
var N = 5;
// Given N points
var arr = [
  [1, 2],

  [2, 1],

  [2, 2],

  [2, 3],

  [3, 2],
];
```

6. Javascript Code to create a login mechanism using object

```
const user = {
  name: 'RELEVEL',
  userName: 'relevel',
  password: 'password:)',
  login: function(userName, password) {
    if (userName === this.userName && password ===
this.password) {
      console.log('Login Successfully');
    } else {
      console.log('Authentication Failed!!');
    }
  },
};
```

```
user.login(‘relevel’, 'relevel');
user.login('relevel', 'password:)');

// Authentication Failed!!
// Login Successfully
```

Explanation for the above program:

Here we have used a object called user and define the below property

a) name – RELEVEL
b) userName – relevel
c) password – password:)
d) login as function

calling the function with passing userName and password parameter the function will check if the userName and password is same or not

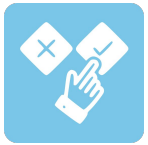
if same log Login Successfully
if not log Authentication Failed!

Explanation for the above program:

- Here we have used a object called car and define the below property

a) Color – black
b) Speed – 120Kmph
c) Brand – Audi
d) Start and stop as function

- calling the property and functionality



MCQs

Q 1 – From the below code what is the output

```
const car = {

    color: 'Black',
    speed: '120Kmph',
    brand: 'Audi',
    start: function () {
        console.log('Car started');
    },
    stop: function () {
        console.log('Car stopped');
    },
}
const newCar = car;
newCar.brand = 'BMW';
console.log(car.brand);
```

A – Audi
B – BMW
C – undefined
D - Error

Q 2 – JSON stand for

A – Java Symbol Object Notation
B – Java Script Object Notation
C – Java Script Object Nation
D - None of the above.

Q 3 – InBuilt function to get the keys of the object

A – Object.keys(obj)
B – Object.values(obj)
C – Object.entries(obj)
D - None of the above.

Q 4 - Which of the following options are correct for accessing the object in Javascript?

```
Const car = { brand: 'Audi' }
```

A- car.brand
B- brand
C- car['brand']
D- All the above

Q 5 – Which of the below function is used to convert string to JSON?

A – JSON.stringify(string)
B – JSON.parse(string)
C – JSON(string)
D – JSON.jsonify(string)

Practice problem

1. Program to demonstrate destructuring in nested objects

1. Program to clone the object and change the property and then iterate the array of objects using inbuilt functions and to console the object property and frame a meaning full sentence.

Upcoming Session

- Object Functions
- Higher-Order Arrays Methods
- Destructuring of objects and Array
- Spread and Rest Operator
- Exercises

Thank you

Topics covered : use icons from the scratchpad on your Figma file

Introduction

Side Effects of Functions

Scopes and Bindings

Closures

Higher Order Functions and Abstraction

Composability

Practice questions