Data Structures: Objects

Relevel
by Unacademy

Concepts

- Introduction: Basics of Objects.
- What are Objects
- JSON
- Properties of JSON objects.
- Difference in JSON Object and Javascript Object.
- Looping through an object.

Concepts

- Different ways for Looping through an object.
- Destructuring in Javascript.
- Array and Object Destructuring.
- Destructuring in Nested Objects.
- Rest in Object Destructuring.
- Few fundamental problems solving:

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

- 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 Constructor and this keyword

```
// 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

There is an inbuilt function for creating an object in javascript

Example:

```
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

```
Dell
Alienware
['SSD', 'HDD']
['256', '512']
8
```

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).

Object Function

1) **object.freeze()**

The **Object.freeze()** method **freezes** an object for preventing the changes. A frozen object can no longer be changed; freezing an object prevents new properties from being added to it, existing properties from being removed, prevents changing the enumerability, configurability, or writability of existing properties, and prevents the values of existing properties from being changed.

```
const obj = {
    prop: 'relevel'
  };

  Object.freeze(obj);

  obj.prop = 'relevels';
  // Throws an error in strict mode

  console.log(obj.prop);
  // expected output: relevel
```

1) object.seal()

The `Object.seal()` method seals an object, preventing new properties from being added to it and marking all existing properties as non-configurable. Values of present properties can still be changed as long as they are writable.

```
const object1 = {
    property1: 'relevel'
  };

  Object.seal(object1);
  object1.property1 = 'relevels';
  console.log(object1.property1);
  // expected output: relevels

  delete object1.property1; // cannot delete when sealed
  console.log(object1.property1);
  // expected output: relevels
```

2) object.isSealed()

The **Object.isSealed()** method determines if an object is sealed.

```
const object1 = {
    property1: 42
  };

  console.log(Object.isSealed(object1));
  // expected output: false

  Object.seal(object1);

  console.log(Object.isSealed(object1));
  // expected output: true
```

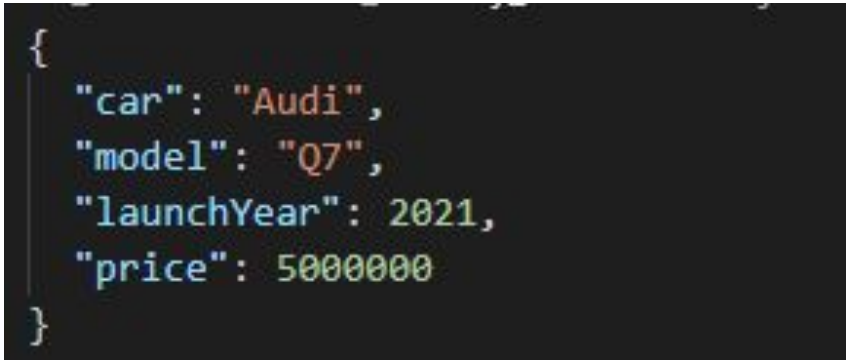
3) object.is()

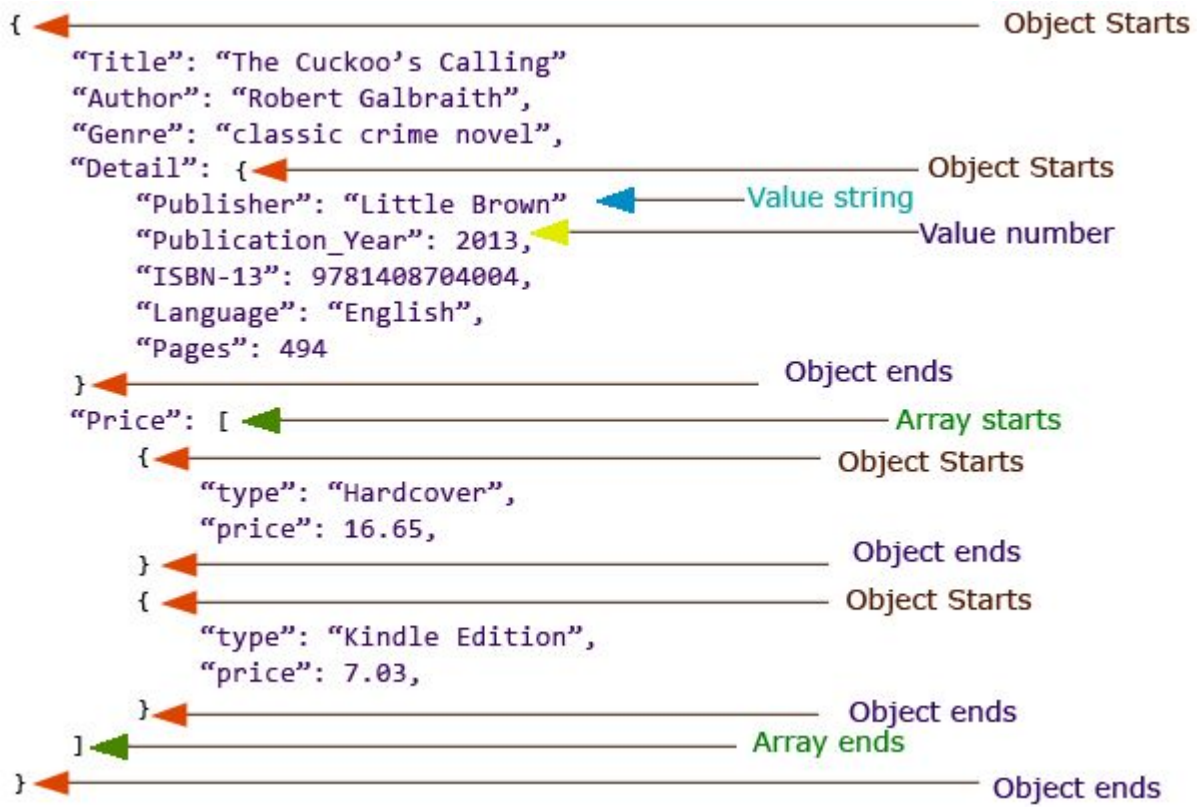
The object.is() method is used to compare two values and return true if both are same

```
Object.is(‘relevel’, 'relevel’’);          // true
```

JSON JavaScript Object Notation

JSON stands for JavaScript Object Notation. It is basically a widely accepted format to exchange data between various application including client-server
applications as well and also a great alternative to XML. The file containing JSON objects is saved with the extension .json.




{
"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

JSON objects can be stored in Javascript objects and can be accessed similarly as we do for Javascript objects using the '.' (dot) operator and object[" "].

```
// JSON Objects

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.

Destructuring in Javascript:

Destructuring in JavaScript is an expression that makes it possible to unpack values from arrays, or properties from objects. We can extract data from arrays and objects and assign them to distinct variables.The value that should be unpacked from the sourced variable is defined on the left-hand side.

Destructuring assignment Using Functions:

```
function getGame(){
    let games = ["Cricket", "Football" , "Hockey", "Golf"];
    return games;
}

let [game_1, game_2] = getGame();

console.log(game_1);//"Cricket"
console.log(game_2);//"Football"
```

Using Default Values:

```
let [game_1 = "Basketball", game_2 = "Golf"] = ["Cricket"];

console.log(game_1);//"Cricket"
console.log(game_2);//"Golf"
```

Object Destructuring

```
let newAvenger = {realName: "Tony Stark", city: "California", heroName: "Iron Man"};

let {realName, city, heroName} = newAvenger;

console.log(realName); //"Tony Stark"
console.log(city); //"California"
console.log(heroName); //"Iron Man"
```

Declaring the variables before destructuring assignment

```
let newAvenger = {realName: "Tony Stark", city: "California", heroName: "Iron Man"};
let realName, city, heroName;

({realName, city, heroName} = newAvenger);

console.log(realName); // "Tony Stark"
console.log(city); //"California"
console.log(heroName); //"Iron Man"
```

Combination of Array and Object Destructuring:

```
let newAvenger = {realName: "Tony Stark", city: ["California", "Malibu"], heroName: "Iron Man"};

let {realName: foo, city: bar} = newAvenger;

console.log(foo); // "Tony Stark"
console.log(bar); // ["California", "Malibu"]
```

Object Destructuring in Nested Objects:

```
let newAvenger = {
    realName: "Tony Stark",
    location: {
        country: "USA",
        city: "California"
    },
    heroName: "Iron Man"
};

let {
    realName: foo,
    location: {
        country : bar,
        city: x
    },
} = newAvenger;

console.log(foo); // "Tony Stark"
console.log(bar); // "USA"
```

Rest in Object Destructuring:

```
let newAvenger = {
    realName: "Tony Stark", country: "USA", city: ["California", "Malibu"], heroName: "Iron Man"
};

let {realName, country, ...restOfDetails } = newAvenger;

console.log(realName); // "Tony Stark"
console.log(restOfDetails); // { city: [ 'California', 'Malibu' ], heroName: 'Iron Man' }
```

Rest Parameters:

```
function getProduct(...input){
    let product = 1;
    for(let item of input){
        product *= item;
    }
    return product;
}

console.log(getProduct(1, 2)); // 2
console.log(getProduct(1, 2, 3, 4)); // 24
```

1) 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!!');
    }
  },
};
```

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

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

```
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}`)
}
```

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

Input: OCCURRENCE

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

```
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")
```

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

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

MCQ

Q 1 – From the below code what is the output

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

Ans: B

```
     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);
```

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.

Ans: B

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.

Ans: A

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

Ans: A and C

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)

Ans: B

Practice problem

1. Program to demonstrate destructuring in nested objects
2. 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

- Pass by Value, Pass by Reference
- Pure/Impure Functions; Side Effects
- Closures
- Higher-Order Functions
- Composability
- Arrow function
- IIFE
- Taking user inputs in JS

Thank you