OBJECTS IN JAVASCRIPT

BY ASHLESHA PATIL

❑ Introduction

- An object in JavaScript is a data structure used to store related data collections. It stores data as key-value pairs, where each key is a unique identifier for the associated value. Objects are dynamic, which means the properties can be added, modified, or deleted at runtime.
- JavaScript is an object-based language. Everything is an object in JavaScript.
- JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

▪ Defination

An object in JavaScript is a collection of key-value pairs.

❑ Creating Objects

- Object Literal Syntax

```
let person = {
    name: "John",
    age: 30,
    fun: function(){
    console.log(function inside object);  }};
```

❑ Accessing Properties

- **Accessing Properties**

  1) Dot Notation: objectName.propertyName

  2) Bracket Notation: objectName["propertyName"]

Example:

```
let person = {
    name: "Alice",
    age: 25 };
console.log(person.name); // Dot notation
console.log(person["age"]); // Bracket notation
```

❑ Modifying Properties

- **Modifying Properties**

1) Add new properties: object.property = value;
2) Update properties: object.property = newValue;
3)Delete properties: delete object.property**;**

Example:

```
const player = {
    pno: 18,
    pname: "Virat",
    mp: 100,
    rs: 10000,
   'run-scored': 6000
};
 player.wife = "Anushka";      // Adding property
delete player.rs;              // Deleting property
player.pno = 81;               // Modifying property
console.log(player);
```

❑ Nested Objects and Function

Nested Objects: Objects can contain other objects or functions as properties.
You can access and manipulate the nested properties using dot notation or bracket notation.

Example

```
const product = {
        name: 'Shirt',
        'delivery-time': '1 day',
      rating: {
         stars: 4.5,
         count: 87  },
      fun: () => {
       console.log('Function inside object');  }
    };
console.log(product.rating.count);        // Access nested property
product.fun();                                    // Call function inside object
```

❑ Shallow Copy and Reference

Understanding References

- **Objects are reference types**

When you assign an object or array to a new variable without creating a copy, both variables point to the same underlying data in memory. This means changes to one will reflect in the other.

Example

```
const obj1 = { message: "Welcome" };
const obj2 = obj1;                    // Reference copy
obj1.message = "Hello";
console.log(obj2.message);            // "Hello“
```

- **Shallow Copy**

A shallow copy creates a new object or array that has the same top-level properties or elements as the original. However, if those properties or elements are themselves objects or arrays, they are still *referenced* (not duplicated). This means changes to nested objects or arrays will affect both the original and the shallow copy.

Example

```
const obj3 = { ...obj1 };          // Shallow copy
obj1.message = "Hi";
console.log(obj3.message);   // "Hello"
```

THANK YOU !!