ES6 ASSIGNMENT-1

1. **Constants:** Declare a constant & confirm its value cannot be changed.

```
<!DOCTYPE html>
<html>
    <head>
        <title>demo</title>
    </head>
    <head>
    <body>
        <h3>constant declaration</h3>
        <script src=index.js>
        </script>

    </body>
    </head>
</html>
```

```
const obj1={
    name:"Rakshita"
};
console.log(obj1.name);
obj1.name="Tomar";
console.log(obj1.name);
```

PS C:\Users\siri\Desktop\node.js\Practise> node index

Rakshita

Tomar

BROWSER OUTPUT:

constant declaration

Rakshita

Tomar

2. **Scoping:** Declare a variable inside if condition & make sure that it is not accessible outside if condition.

```
<!DOCTYPE html>
<html>
```

```
    <head>
        <title>demo</title>
    </head>
    <head>
    <body>
        <h3>Variable declaration inside if block</h3>
        <script>
            var a = 1;
            var b = 2;
            if(a===1){
                var a=20;
                var b=10;
                document.write(a,"<br>");
                document.write(b,"<br");
            }
        </script>
    </body>
    </head>
</html>
```

3. **Enhanced object properties:** Create an 'Order' object having data members 'id', 'title', 'price'. Add the methods printOrder() & getPrice(). Now, copy the order object using Object.assign().

```
<!DOCTYPE html>
<html>
    <head>
        <title>demo</title>
    </head>
    <body>
        <h3>Enhanced Object Properties</h3>
        <script>
            let Order = {
                id:100,
                title:"Rakshita",
                price:3000,
                printOrder(){
                    document.write(this.id);
                    document.write(this.title);
                    document.write(this.price)
                },
                getPrice(){
                    document.write(this.id);
                    document.write(this.title);
                    document.write(this.price)
                },
            }
            const obj1 = {};
```

```
            const obj2 = Object.assign({},Order);
            document.write(obj2);
        </script>
    </body>
</html>
```

4. **Arrow functions:** Take an array of strings & convert it into another array of object which has two properties {string, string_length}. For example:

   let names = ['Tom', 'Ivan', 'Jerry']

   Output: [{name: 'Tom', length: 3}, {name: 'Ivan', length: 4 }, {name: 'Jerry', length: 5}]

```
<!DOCTYPE html>
<html>
    <head>
        <title>demo</title>
    </head>
    <body>
        <h3>Arrow functions</h3>
        <script>
            let array = ['Tom','Ivan','Jerry'];
            let newArr = array.map(item => {
                    return {
                        'name': item,
                        'length': item.length
                    }
                })
                document.write(newArr);
        </script>
    </body>
</html>
```

5. **Extended parameter handling:**
   a. Write a add() with default values.
   b. Write a function userFriends() that takes 2 arguments username & array of user friends. The function should print username & his list of friends. (Use rest parameters)
   c. Write a function printCapitalNames() that takes five names as argument & prints them in capital letters. Use spread operator in order to call printCapitalNames() function.

```html
<!DOCTYPE html>
<html>
    <head>
        <title>demo</title>
    </head>
    <body>
        <h3>Extended parameter handling</h3>
        <script src=index.ts></script>
        </script>
    </body>
</html>
```

```javascript
let a = function(value=10)
{
    return value;
}
console.log('Default parameter is ' + (a()));

// b)
let userFriends = function(username, ...userfriends){
    this.username = username;
    this.userfriends = userfriends;
    console.log(username,userfriends);
}
userFriends("Rakshita","Sandhya","Ritu","Riya");

//c)

const names =["rakshita","sandhya","aditya","ritu","riya"];
let printCapitalNames = function(){
    const x = names.map(function(name){
        return name.toUpperCase();
    })
    console.log(x);
}
printCapitalNames();
```

6. **Template literals:** Draft a ticket to Sysnet that describes problem with your laptop. Use 'template literals' to add value of laptop model, your desk no, your name etc.

```html
<!DOCTYPE html>
<html>
    <head>
        <title>demo</title>
    </head>
    <body>
        <h3>Template literals</h3>
        <script src=index.ts></script>
```

```
        </script>
    </body>
</html>
```

```
const names = "rakshita";
const deskno = 100;
const laptopmodel = "hp";
const message = "Hello SYSNET, My name is ${name}, desk number is ${deskno},
and laptop model is ${hp}."
"Here i am facing some issue to install the softwares it is asking admin
rights to install, will u please resolve this error."

"Thanks and regards"
"Gowthami"

console.log(message);
```

7. **De-structuring assignment:**
    a. Suppose there is a javascript array with 4 elements. Print the value of 3rd element using array matching.
    b. Create an organization object having attributes name, address. Write a program to retrieve pin code of an address using object deep matching.

```
<!DOCTYPE html>
<html>
    <head>
        <title>demo</title>
    </head>
    <body>
        <h3>Destructuring Assignment</h3>
        <script src=index.ts></script>
        </script>
    </body>
</html>
```

7-a)

```
let friends = [“rakshita”,”sandhya”,”ritu”,”riya”];
let[item1,item2,item3,item4] = friends;
console.log(item3);
```

7-b)

8. **Classes & Modules:** Write a class Account with attributes id, name, balance. Add two sub classes SavingAccount & CurrentAccount having specific attribute interest & cash_credit respectively. Create multiple saving & current account objects. Write a functionality to find out total balance in the bank.

```
<!DOCTYPE html>
<html>
    <head>
        <title>demo</title>
    </head>
    <body>
        <h3>Classes & Modules</h3>
        <script src=index.ts></script>
        </script>
    </body>
</html>
```

```
class Account{
    constructor(id,name,balance)
    {
        this.id = id;
        this.name = name;
        this.balance = balance;
    }
}
class SavingsAccount extends Account{
    constructor(id,name,balance,interest)
    {
        super(id,name,balance);
        this.interest = interest;
    }
    totalBalance = () => { this.balance += this.interest;
        console.log(this.balance);};
}
class CurrentAccount extends Account{
    constructor(id,name,balance,cashCredit)
    {
        super(id,name,balance);
        this.cashCredit = cashCredit;
    }
    totalBalance = () => { this.balance += this.cashCredit;
        console.log(this.balance);};
}
```

```
let S1 = new SavingsAccount(1001,'rakshita',40400,4000);
let S2 = new CurrentAccount(10003, 'sandhya', 30000,3800);
S1.totalBalance();
S2.totalBalance();
```