ES6 Assignments

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

```
const str = "hello";
str = "hi";
console.log(str);
```

```
>> Cannot assign to 'str' because it is a constant.
```

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

```
if (true) {
    let a = 10;
}
console.log(a);
```

```
>> Cannot find name 'a'.
```

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().

```
function Order(id, title, price){
  this.id = id;
  this.title = title;
  this.price = price;
}

order1 = new Order(1,"abc",100);
order2 = new Order();
var order2 = Object.assign(order1);

getPrice = function(){
  console.log(order1.price);
}();

printOrder = function(){
  console.log(order1.id+" "+ order1.title+" "+order1.price);
  console.log(order2.id+" "+ order2.title+" "+order2.price);
}();
```

```
>> 100
>> 1 abc 100
>> 1 abc 100
```

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)]

```
let names = ["Tom", "Ivan", "Jerry"]

let arrObject = (names) => {
  let arr = [];
  for (let n of names){
    arr.push({name: n,
      length: n.length});
  }
  console.log(arr);
}

arrObject(names);
```

```
>>
[{"name":"Tom","length":3},{"name":"Ivan","length":4},{"name":
"Jerry","length":5}]
```

5. **Extended parameter handling:**

   a. Write a add() with default values.

```
function add(a, b) {
  return a + b;
}

let x = 10, y = 5;
console.log(add(x,y));
```

```
>> 15
```

   b. Write a function user Friends() that takes 2 arguments username & array of user friends. The function should print username & his list of friends. (Use rest parameters)

```
function userFriends(username, ...args){
  return username+ ","+ args;
}

console.log(userFriends("abc", "xyz", "pqr"));
```

```
>> abc,xyz,pqr
```

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.

```
const names = ['Ali', 'Atta', 'Alex', 'John', 'Amy'];

const printCapitalNames = names.map(name =>
name.toUpperCase());

console.log(printCapitalNames)
```

```
>> ["ALI","ATTA","ALEX","JOHN","AMY"]
```

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.

```
let laptop_model = "LENOVO ideapad 310";
let desk_no = "101";
let name = "abc";

console.log(`I am ${name} facing difficulty with
${laptop_model} and my desk number is ${desk_no}.`);
```

```
>> I am abc facing difficulty with LENOVO ideapad 310 and my
desk number is 101.
```

7. **De-structuring assignment:**

a. Suppose there is a javascript array with 4 elements. Print the value of 3rd element using array matching.

```
Array.prototype.diff = function(arr) {
  var ret = [];
  for(var i in this) {
      if(arr.indexOf(this[i]) > -1){
          ret.push(this[i]);
      }
  }
  return ret;
};

var array1 = ["cat", "bat","rat", "ant"];
var array2 = ["gut" ,"dog","rat", "hut"];

console.log(array1.diff(array2));
```

>> ['rat']

b. Create an organization object having attributes name, address. Write a program to retrieve pin code of an address using object deep matching.

```
let organisation1 = {
  name: "John",
  address: {
      street: "Rammurthy nagar",
      city: "Bangalore",
      pincode: 560016
  }
}

let organisation2 = {
  name: "Robert",
  address: organisation1.address
}

console.log((organisation1.address.pincode ===
organisation2.address.pincode));
console.log(organisation1.name === organisation2.name);
```

```
>> true
>> false
```

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

```
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() {
    let newBalance = this.balance * this.interest;
    this.balance = this.balance + newBalance;
    return this.balance;
  }
}
```

```
class CurrentAccount extends Account {
  constructor(id, name, balance, cash_credit) {
    super(id, name, balance);
    this.cash_credit = cash_credit;
  }
  totalBalance() {
    let newBalance = this.balance * this.cash_credit;
    this.balance = this.balance + newBalance;
    return this.balance;
  }
}

var saving = new SavingsAccount("11110001", "abc", 2000, 1.5);
var current = new CurrentAccount("22220001", "xyz", 5000,
0.5);
console.log(saving.totalBalance());
console.log(current.totalBalance());
```

```
>> 5000
>> 7500
```