Objects

Confusing questions:

```
const func = (function(a){
  delete a;//delete keyword only used when we want to delete
           //properties from object.It will not work here.a will return the output
   return a;
})(5);
console.log(func)
```

```
const user = {
   name: "Lazy Coder",
   age: 24,
   `like this video`:true;//to use a properties name with space we have to wrap it with a double quote
};
console.log(user[`like this video`])//we have to access the properties like that with a double quote
```

To use a variable as a property name in an object look at this code:

```
const property = "firstName";
const name = "Hayat Hossain";

const user = {
   [property]: name,
}

console.log(user.firstName);
```

If we want to loop through an object:

```
const user = {
    name: "Hayat Hossain",
    age: 24,
    status: true
};
for (key in user) {
    console.log(key);
    console.log(user[key]);
}
```

Question-01: What's the Output?

```
const obj = {
  a: "one",
  b: "Two",
  a: "three",
};

console.log(obj)
//output:Object { a: "three", b: "Two" }
//so the same name property(a) will always replace by the last value
```

Question-02: Create a function multiplyByTwo(obj) that multiplies all numeric property values of nums by 2

```
let nums = {
    a: 100,
    b: 200,
    title: "My nums",
};
```

```
multiplyByTwo(nums);

function multiplyByTwo(obj) {
    for (key in obj) {
        if (typeof obj[key] === "number") {
            obj[key] *= 2;
        }
    }
}
console.log(nums)//output: Object { a: 200, b: 400, title: "My nums" }
```

Question-03: What's the Output of the following code?

```
const a ={};
const b = {key:"b"};
const c = {key:"c"};

a[b] = 123;
a[c] = 456;

console.log(a[b]);

Answer:
const a = {};
const b = { key: "b" };
const c = { key: "c" };

a[b] = 123;
a[c] = 456;

console.log(a) // output: Object { "[object Object]": 456 }
// we can't directly put an Object as a key into another object.as a result it shows [object object] and
// it is used as a key.so at first value 123 assigned and finally it is replaced by the 456 value in
// "[object object]" value
console.log(a[b]);//output:456
```

Question-04: What's the Output?

```
const user = {name: "Hayat Hossain", age:24}
const admin = {admin:true,...user};
console.log(admin)//output:Object { admin: true, name: "Hayat Hossain", age: 24 }
```

Question-05: What's the Output?

```
const settings = {
    username: "Hayat",
    level: 20,
    health: 90
}
const data = JSON.stringify(settings,["level", "health"]);
console.log(data)//output: {"level":20,"health":90}
```

Question-06: What's the output?

```
const shape = {
    radius: 10,
    diameter() {
        return this.radius * 2;
    },
    perameter: () => 2 * Math.PI * this.radius,
};
console.log(shape.diameter())//output: 20
console.log(shape.perameter());//output: NaN
```

Question-07: What is destructuring in objects?

```
let user = {
    name: "Hayat",
    age: 24,
    fullName: {
        first: "Hayat Hossain",
        last: "Chowdhury",
    },
};

const name = "Nakib";
const { name } = user;//output:Uncaught SyntaxError: redeclaration of const name
//so to overcome this error we have to rename it like this if a similar name variable exists
const { name: Myname } = user
console.log(Myname)
//nested destructuring
const { fullName: { first } } = user;
console.log(first)
```

Question-08: What's the output?

```
let c = {Greeting: "Hey!"};
let d;

d=c;
c.Greeting = "Hello";
console.log(d.Greeting);//Output: "Hello"
//so when a object is copied to another varible it's just
//simply refference the object location.so changing one will
//change the other variable as well
```

Question-09: What's the output?

Here both are independent object and located different place in memory.

```
console.log({ a: 1 } == { a: 1 }) //output: false
console.log({ a: 1 } === { a: 1 });//output:false
```

Question-10: What's the output?

```
let person = {name:"Hayat"};
const members = [person];
//person = null;

console.log(members);//output:Array [ {…} ]0: Object { name: "Hayat" }
console.log(person)//null

//if we change the properties

person.name = null
console.log(members);//output: Object { name: null }

//changing properties of an object will affect the output
```

Question-11: What's the output?

```
const value = { number: 10 };
const multiply = (x = { ...value }) => {
    console.log((x.number *= 2))
};

multiply();//output: 20
multiply();//output: 20// when we didnot pass argument as (value) the spread operator clone the value object and create new value object
multiply(value);//output: 20
multiply(value);//output: 40//but when we pass the value as argument it takes the reference of the value(memory location) and modify it so
```

Question-12: What’s the output?

```
function changeAgeAndReference(person) {
    person.age = 25;//here person parameter have the reference of personObj1.so it will chage the age of
    //personObj1 object to 25
    // here we reassigning the person variable to other object so this will not affect the reference(personObj1)
    person = {
        name: "Hayat",
        age: 50,
    };
    return person;
}
const personObj1 = {
    name: "Hossain",
    age: "24"
};
const personObj2 = changeAgeAndReference(personObj1);
console.log(personObj1);//output: Object { name: "Hossain", age: 25 }
console.log(personObj2);//outpur: Object { name: "Hayat", age: 50 }
```

Question-13: What’s Shallow copy and Deep copy?

When one object holds the reference to another object this is called shallow copy.

when an object is completely clone to another variable that is called a deep copy

```
let user = {
    name: "Hayat Hossain",
    age: 24
}

const objectClone = Object.assign({}, user);
const objectClone1 = JSON.parse(JSON.stringify(user));
const objectClone2 = { ...user };
objectClone.name = "sakib"
objectClone1.name = "akib"
objectClone2.name = "nakib"

console.log(user, objectClone, objectClone1, objectClone2)
//output:
//Object { name: "Hayat Hossain", age: 24 }

//Object { name: "sakib", age: 24 }

//Object { name: "akib", age: 24 }

//Object { name: "nakib", age: 24 }
```