```javascript
console.clear();

/*

=====================================================================

                Object Interview Questions 🚀

=====================================================================

*/

/*

1. Write a function that compares two objects to determine if they have the same properties and values.

*/

let obj1 = { name: "Vinod", age: 30, isStudent: false };

let obj2 = { name: "Vinod", age: 30, isStudent: false, country: "India" };

const compareObjects = (obj1, obj2) => {

    if (Object.keys(obj1).length !== Object.keys(obj2).length) {

        return false;
    }

    for (let key in obj1) {

        if (obj1[key] !== obj2[key]) {

            return false;
        }
    }

    return true;
}

console.log(compareObjects(obj1, obj2));

/*

2. Given an object representing a student, write a function to add a new subject with it's corresponding grade to the student's record. Also check if the grade property exists.

*/

let studentData = {

    name: "Bobby",

    age: 21,

    grades: {

        Maths: 90,

        Science: 85,

        History: 88
    }
}

const addSubject = (student, subject, marks) => {

    if (!student.grades) {

        student.grades = {};
    }

    return (student.grades[subject] = marks);
}

```

```
addSubject(studentData, "English", 95);

console.log(studentData);

let employeeData = {

    name: "John",

    age: 30,

    personalInfo: {

        address: "123 Main St",

        phone: "555-555-5555"
    }
}

// adding email property in the personalInfo object

const addEmail = (employee, email) => {

    return employee.personalInfo.email = email;
}

addEmail(employeeData, "i8o9g@example.com");

console.log(employeeData);

// deleting email property from the personalInfo object

const deleteEmail = (employee) => {

    delete employee.personalInfo.email;
}

deleteEmail(employeeData);

```

```
console.log(employeeData);

/*

3. Write a function to clone an object (shallow copy).

*/

const cloneObject = (obj) => {

    return { ...obj };
}

let originalObject = {

    name: "Alice",

    age: 25,

    adress: {

        city: "New York",

        state: "NY"

    }
}

// If your object has nested objects, then spreading ({ ...obj }) only makes a shallow copy. That means nested references are shared between original and clone.

let clone = cloneObject(originalObject);

console.log(clone);

clone.adress.city = "San Francisco";

console.log(clone);

console.log(originalObject);

/*

4. Merge two objects. If both have the same key, the second object should overwrite.

*/

const mergeObjects = (obj1, obj2) => {

    return { ...obj1, ...obj2 };
}

const firstObj = { x: 1, y: 2 };

const secondObj = { y: 3, z: 4 };

console.log(mergeObjects(firstObj, secondObj));

/*

5. Count the number of properties in an object.

*/

const countProperties = (obj) => {

    return Object.keys(obj).length;
}

console.log("The number of properties (keys) in the object is ", countProperties({ x: 1, y: 2, z: 3 }));

/*

6. Check if a property exists in an object.

*/
```

```

const hasProperty = (obj, key) => {

    return obj.hasOwnProperty(key)
}

let user = {

    id: 1,

    username: "John Doe"
}

console.log(hasProperty(user, "id"));

console.log(hasProperty(user, "email"));

/*

7. Convert an object to an array of key-value pairs.

*/

const ObjectToPairs = (obj) => {

    return Object.entries(obj);
}

console.log(ObjectToPairs({ a: 1, b: 2, c: 3 }));

/*

8. Create a function that removes a specific key from an object.

*/

const removeKey = (obj, key) => {

    delete obj[key];

    return obj;
}

let item = { id: 1, name: "IPhone", price: 100000 };

console.log(removeKey(item, "id"));

/*

9.  Iterate over all keys and values in an object.

*/

const printObject = (obj) => {

    for (let [key, value] of Object.entries(obj)) {

        console.log(`${key}: ${value}`);
    }
}

let userProfile = { name: "Sara", Profession: "Software Engineer" };

printObject(userProfile);

/*

10. Get only keys or values from an object.

*/

let sampleObj = { a: 10, b: 20, c: 30 };

console.log(Object.keys(sampleObj));

console.log(Object.values(sampleObj));
```

```
/*

11. Convert an object to a string.

*/

const objectToString = (obj) => {

    return JSON.stringify(obj);
}

console.log(objectToString({ name: "John", age: 25, city: "New York" }));

/*

12. Convert a string to an object.

*/

const stringToObject = (str) => {

    return JSON.parse(str);
}

console.log(stringToObject('{"name": "John", "age": 25, "city": "New York"}'));

/*

13. Check if an object is empty.

*/

const isEmptyObject = (obj) => {

    return Object.keys(obj).length === 0;
}

```

```
console.log(isEmptyObject({}));

console.log(isEmptyObject({ name: "John", age: 25 }));

/*

14. Get the first key in an object.

*/

const getFirstKey = (obj) => {

    return Object.keys(obj)[0];
}

console.log(getFirstKey({ a: 1, b: 2, c: 3 }));

/*

15. Get the last key in an object.

*/

const getLastKey = (obj) => {

    return Object.keys(obj)[Object.keys(obj).length - 1];
}

console.log(getLastKey({ a: 11, b: 22, c: 33 }));

/*

16. Get the first value in an object.

*/

const getFirstValue = (obj) => {

    return Object.values(obj)[0];
}

console.log(getFirstValue({ a: 100, b: 200, c: 300 }));

/*

17. Get the last value in an object.

*/

const getLastValue = (obj) => {

    return Object.values(obj)[Object.values(obj).length - 1];
}

console.log(getLastValue({ a: 1000, b: 2000, c: 3000 }));

/*

18. Get the first key and value in an object.

*/

const getFirstKeyValuePair = (obj) => {

    return Object.entries(obj)[0];
}

console.log(getFirstKeyValuePair({ a: 1, b: 2, c: 3 }));

/*

19. Get the last key and value in an object.

*/

const getLastKeyValuePair = (obj) => {

    return Object.entries(obj)[Object.entries(obj).length - 1];
}

console.log(getLastKeyValuePair({ a: 10000, b: 20000, c: 30000 }));

/*

20. Get the sum of all values in an object.

*/

const getSumOfValues = (obj) => {

    return Object.values(obj).reduce((acc, currentValue) => acc + currentValue, 0);
}

console.log(getSumOfValues({ a: 10, b: 20, c: 30 }));

/*

21. Get the average of all values in an object.

*/

const getAverageOfValues = (obj) => {

    return Object.values(obj).reduce((acc, currentValue) => acc + currentValue, 0) / Object.values(obj).length;
}

console.log(getAverageOfValues({ a: 10, b: 20, c: 30 }));

/*

22. Get the maximum value in an object.

*/

```

```
const getMaxValue = (obj) => {

    return Math.max(...Object.values(obj));
}

console.log(getMaxValue({ a: 10, b: 20, c: 30 }));

/*

23. Get the minimum value in an object.

*/

const getMinValue = (obj) => {

    return Math.min(...Object.values(obj));
}

console.log(getMinValue({ a: 10, b: 20, c: 30 }));

/*

24. Get the length of the longest key in an object.

*/

const getLongestKey = (obj) => {

    return Math.max(...Object.keys(obj).map((key) => key.length));
}

console.log(getLongestKey({ a: 10, b: 20, c: 30 }));

/*

25. Get the length of the shortest key in an object.

*/
```

```

const getShortestKey = (obj) => {

    return Math.min(...Object.keys(obj).map((key) => key.length));
}

console.log(getShortestKey({ a: 10, b: 20, c: 30 }));
```