# "Objects Useful Methods"

eg:

```
const product = {
    id : 1,
    name : "Laptop",
    category : "Computers",
    brand : "Example Brand",
    price : 999.99,
    stock : 50,
    description :
        "Powerful laptop with quad core i5 proc",
    image : "image link",
};
```

1) Object.keys() :- Returns an array containing the names of all enumerable own properties of an object.

let keys = Object.keys(product);

↳ [ 'id', 'name', 'category', 'brand', ... ]

Got an array containing all the keys of object product.

2) Object.values() : returns an array containing the values of all enumerable own properties of an object.

> console.log ( Object.values (product)); → similarly here we get an array contain values

3) Object.entries() : returns an array containing arrays of key-value pairs for each enumerable own property of an object.

console.log ( Objects.enteries (product));

↳ [
[ 'id', 1 ],
[ 'name', 'Laptop' ],
[ 'category', 'computers' ],
...
]

return an array containing ~~array~~ entry array which have key, value as elements

4) Object.hasOwnProperty() : returns a boolean indicating ~~wh~~ whether the object has the specified property as an own property.

eg :- console.log ( product.hasOwnProperty ("name")); // true
console.log ( product.hasOwnProperty ("isStudent")); // false.

5. Object.assign(): copies the value of all enumerable own properties from one or more source objects to a target object.

```
const target = {a: 1, b: 2};
const source = {b: 3, c: 4};
const mergedObject = Object.assign(target, source);
console.log(mergedObject);
// output: {a: 1, b: 3, c: 4}
```

6. Object.freeze(): Freezes an object, preventing new properties from being added to it and existing properties from being modified or deleted.

eg
```
Object.freeze(product);
product.id = "5561";
console.log(product);
```
⇒ we will see there is no change in id

---

## Questions - Objects

1. what will be the output?

```
const target = {a: 1, b: 2};
const source = {b: 3, c: 4};
const merged = Object.assign({}, target, source)
console.log(merged);
```

output → {a: 1, b: 3, c: 4} How?

as first target assigned to {}, now, src is assigned to target & b: 3 as it overwrites.

2. Given an object representing a student, write a function to add a new subject with its corresponding grade to the student's record. Also check if the grades property is present or not?

```
let student = {
    name : "Bob",
    grades : {
        maths : 90,
        science : 88,
        history : 88,
    },
};
```

~~function~~ addSubjectGrade(student, "computer", 92);
console.log(student);

So, function addSubjectGrade(student, subject, grade) {

student.grades.subject = grade; ✗

↳ grade: {---, subject: 92}. ✗

```
if(!student.grades) {
    student.grades = {};
}
return (student.grades[subject] = marks);
}
```

Q2) Write a function that compares two objects to determine if they have the same ~~value~~ properties and values. (we can't compare objects, but we can do its properties).

⇒. (1) we can check whether the length of both the objects are same or not.

i.e. obj1.length == obj2.length ✗

there is no length property in Object itself.

Array, string are objects as they have length property & iterable. But objects itself don't have.

(2) what can we do?. we can get keys, values array then we can compare length.

```
let o1 = Object.keys(obj1);
let o2 = Object.keys(obj2);
if(o1.length != o2.length) return false.
```

Now check whether the key of both are same or not.
We will use **for-in loop** as object is not iterable

for-of loop works in array, strings, But not on object itself

```
for (let key in obj1) {
    if (obj1[key] !== obj2[key]) { return false; }
}
return true.
```

Q3 Write a function that transforms an array of an object into an object where the keys are the objects' ids.

```
let inputArray = [
    { id: 1, name: "Alice" },
    { id: 2, name: "Bob" },
    { id: 3, name: "Charlie" },
];
```

Output :- { '1': { id: 1, name: "Alice" }, ... }

⇒

```
let obj = {};
for (let key of arr) {
    obj[key.id] = key;
}
return obj;
```