CHEAT SHEET FOR OBJECT

➢ Addind new properties :

We can add new properties to an existing object by simply giving it a value. Assume that the student object already exist and we are adding email property in it.

```
Ex:   const student = {
            name: “Student 1”,
            roll_no: 101,
            city: “Surat”
      }
      student.email = “student.1@bacancy.com”;
      console.log(student);

O/P:  {
            name: “Student 1”,
            roll_no: 101,
            city: “Surat”,
            email: “student.1@bacancy.com”
      }
```

➢ Deleting properties :

The delete keyword used to delete property from object.

```
Ex:   const student = {
            name: “Student 1”,
            roll_no: 101,
            city: “Surat”,
            email: “student.1@bacancy.com”
      }
      delete student.email;
      console.log(student);

O/P:  {
            name: “Student 1”,
            roll_no: 101,
            city: “Surat”,
      }
```

➢ Object.assign() :

The Object.assign() method copies all enumerable own properties from one or more source objectsto a target object. It returns the modified target object.
In below output of both object obj1 and finalObj will be same.

Ex: const obj1 = { name: “Manish” id: 101 },
const obj2 = { id: 102, email: “manish.rathod@bacancy.com” }

const finalObj = Object.assign(obj1, obj2);
console.log(finalObj);

O/P: {
name: “Manish”,
id: 102,
email: “manish.rathod@bacancy.com”
}

- ## Object.create() :

Object.create() method create a new object, using an existing objects as the prototype of the newly created object.

Ex: const obj1 = { a: “Manish”, b: 101 };
const obj2 = Object.create(obj1);
obj2.name = “Manish”;
obj2.city = “Surat”;
console.log(obj2);

O/P: { name: “Manish”, city: “Surat” }

Hence name and city property will be assign to onlt newly created object names obj2.

- ## .keys() :

The Object.keys() method returns an array of all keys present in object.

Ex: let obj = { name: “Manish”, id: 101 };
console.log(Object.keys(obj));

O/P: [“name”, “id”]

- ## .values() :

The Object.values() method returns an array of all values present in object.

Ex: let obj = { name: “Manish”, id: 101 };
console.log(Object.values(obj));

O/P: [“name”, 101]

.entries() :

The Object.entries() method returns an array of given object's own enumerable string-keyed property [key, values] pairs.

```
Ex:   let obj = {name: "Manish", id: 101};
      for( [key, value] of Object.entries(obj)){
            console.log(key, value);
      }
```

```
O/P:  name - Manish
      id - 101
```

.getOwnPropertyNames() :

The Object.getOwnPropertyNames() method returns an array of all peroperties (including non-enumerable (properties which we can't access using iteration.) properties.) found directly in a given object and returns all properties in form of array.

```
Ex:   let obj = { name: "Manish", id: 101 };
      console.log(Object.getOwnPropertiesNames(obj));
```

```
O/P:  ["name", "id"]
```

.freeze() :

The Object.freeze() method freezes the object. Freezes objects are those which are inmutable, means we can not add any new property into the objects or we can not remove any properties from the object.

```
Ex:   let obj = { name: "Manish", id: 101 }
      Object.freeze(obj);
      obj.id = 102; //This line will thorw error message.
      Console.log(obj.id);
```

```
O/P:  101
```

.seal() :

The Object.seal() method is similar to Object.freeze() method. The only difference between Object.seal() and Object.freeze() is that Object.seal() allow to change the value of properties.

```
Ex:   let obj = { name: "Manish", id: 101 }
      Object.seal(obj);
      obj.id = 102;
      Console.log(obj.id);
```

O/P: 102

➢ .fromEntries() :

The Object.fromEntries() method transforms a list of key-value pairs in to object.

Ex:
```
let pairArr = [ [“name”, “Manish”], [“id”, 101] ];
let arrrToObj = Object.fromEntries(pairArr);
console.log(arrrToObj);
```

O/P: { name: “Manish”, id: 101 }

➢ .getPrototypeOf() :

The Object.getPrototypeOf() method returns the prototype of the specified object.

Ex:
```
let obj = { name: “Manish”, id: 101 }
Object.prototype.display = function(){
        return `My name is ${this.name} and my id is: ${this.id}.`;
}
console.log(Object.getPrototypeOf(obj));
```

O/P: {display: [Function (anonymous)}

➢ .is() :

The Object.is() method check if both value are same or not and retur boolean true if both value are same else it return false. It accept two parameter that are going to be compare.

Ex:
```
let obj1 = { name: “Manish”, id: 101 }
let obj2 = { name: “Manish”, id: 102 }
let obj3 = obj1;
console.log(Object.is(obj1, obj2);
console.log(Object.is(obj1, obj3);
```

O/P:
```
false
true
```

➢ .isFrozen() :

The Object.isFrozen() method check if object is frozen or not and return boolean value.

Ex: let obj = { name: “Manish”, id: 101 }
console.log(Object.isFrozen(obj));
Object.freeze(obj);
console.log(Object.isFrozen(obj));

O/P: false
true

- ## .isSeal() :

The Object.isSeal() method check if object is selaed or not and return boolean value.

Ex: let obj = { name: “Manish”, id: 101 }
console.log(Object.isSeal(obj))
Object.seal(obj);
console.log(Object.isSeal(obj))

O/P: false
true