Objects and its internal representation in Javascript:

- ❖ In JavaScript, an object is an unordered collection of key-value pairs. Each key-value pair is called a property.
- ❖ The key of a property can be a string. And the value of a property can be any value, e.g., a string, anumber, an array, and even a function.

For example, the following creates a new person object:

- To create an object with properties, you use the key:value within the curly braces.

```
var person = {
                firstName: “guvi”
                 lastName: “zen”
              };
```

The person object has two properties firstName and lastName with the corresponding values ‘guvi’ and ‘zen’

- When an object has multiple properties, you use a comma (,) to separate them like the above example.

Accessing properties:

To access a property of an object, you use one of two notations: the dot notation and array-like notation.

1) The dot notation (.)

The following illustrates how to use the dot notation to access a property of an object:

objectName.propertyName

to access the firstName property of the person object, you use the following expression:

Person.firstname

This example creates a person object and shows the first name and last name to the console:

```
var person={
        firstname: “guvi”
        lastname: “zen”
        };
console.log(person.firstname);
console.log(person.lastname);
```

2) Array-like notation ([])

★ The following illustrates how to access the value of an object’s property via the array-like notation:

Objectname[‘property name’]

```
var person = {
        firstName: “guvi”
        lastName: “zen”
      };
console.log(person[firstname]);
console.log(person[lastname]);
```

★ When a property name contains spaces, you need to place it inside quotes. For example, the following address object has the 'building no' as a property:

```
let address = {
        'building no': 3960,
        street: 'North 1st street',
        state: 'CA',
        country: 'USA'
        };
```

To access the 'building no' property, you need to use the array-like notation:

```
address['building no']
```

➔ If you use the dot notation, you'll get an error:

```
address.'building no';

SyntaxError: Unexpected string
```

Note that it is not a good practice to use spaces in the property names of an object.

➔ Reading from a property that does not exist will result in an undefined

For example:

```
console.log(address.district);
```

Output:Undefined

3) Modifying the value of a property

➢ To change the value of a property, you use the assignment operator (=).

For example:

```
var person = {
                    firstName: "guvi",
                    lastName: "zen",
                };
person.firstName = 'john';

console.log(person);
```

Output: { firstName: '"john", lastName: "zen"}

In this example, we changed the value of the firstName property of the person object from guvi to john.

4) Adding a new property to an object

➢ Unlike objects in other programming languages such as Java and C# you can add a property to an object after object creation.

The following statement adds the age property to the person object and assigns 25 to it:

person.age=25

5) Deleting a property of an object

- To delete a property of an object, you use the delete operator:

delete objectName.propertyName;

The following example removes the age property from the person object:

```
Delete.person.age;
```

6) Checking if a property exists

- To check if a property exists in an object, you use the in operator:

propertyName in objectName;

The in operator returns true if the propertyName exists in the objectName.

The following example creates an employee object and uses the in operator to check if the ssn and employeeId properties exist in the object:

```
var employee = {
                    firstName: 'guvi',
                    lastName: 'zen',
                    employeeId: 1
                  };
console.log('ssn' in employee);
console.log('employeeId' in employee);
```

Output:

```
false
True
```

Summary

- An object is a collection of key-value pairs.
- Use the dot notation (.) or array-like notation ([]) to access a property of an object.
- The delete operator removes a property from an object.
- The in operator check if a property exists in an object.