Objects keys,values,entries

In JavaScript, objects are collections of key-value pairs, where each key is a string (or Symbol) and each value can be any data type. Looping through objects involves accessing these keys, values, or both, to perform operations on the object's data.

Keys, Values, and Properties:

Properties:
The fundamental building blocks of an object. Each property consists of a key (also known as a property name) and a corresponding value.

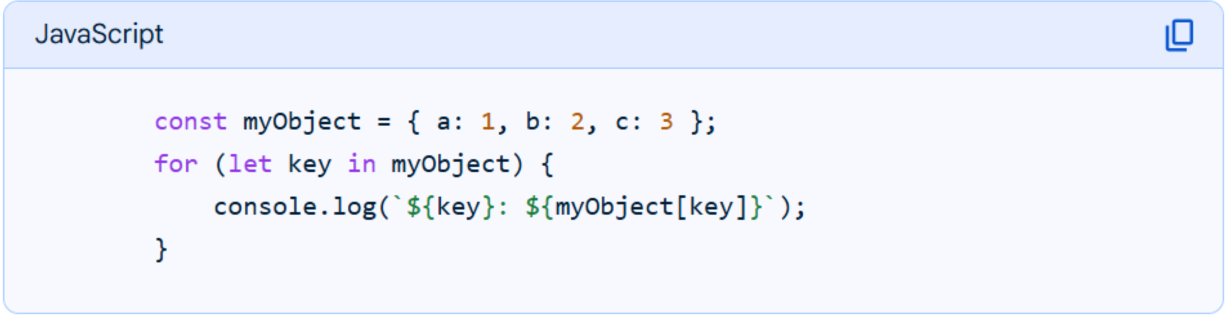
Keys:
The unique identifiers or names associated with a property's value. They are typically strings.

Values:
The data associated with a specific key within an object. Values can be numbers, strings, booleans, arrays, other objects, functions, etc.

Methods for Looping Through Objects:

Several methods facilitate looping through object properties:

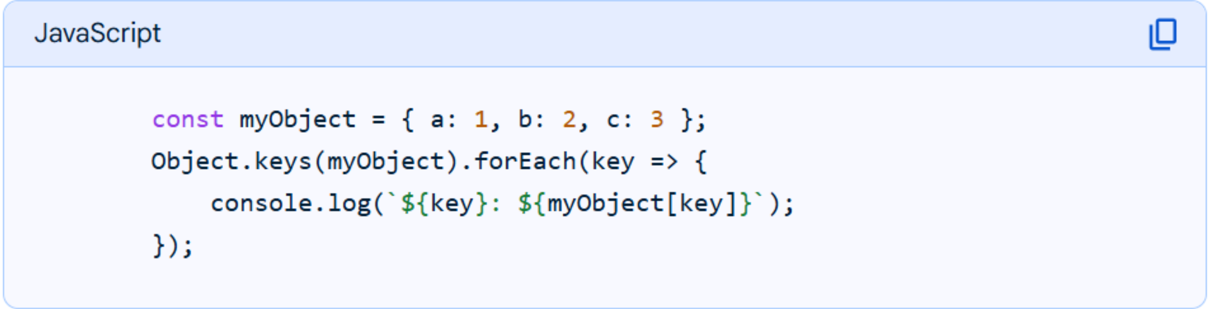
- `for...in` loop:
  - This traditional loop iterates over the enumerable properties of an object, including inherited ones from the prototype chain.
  - Example:



```
const myObject = { a: 1, b: 2, c: 3 };
for (let key in myObject) {
    console.log(`${key}: ${myObject[key]}`);
}
```

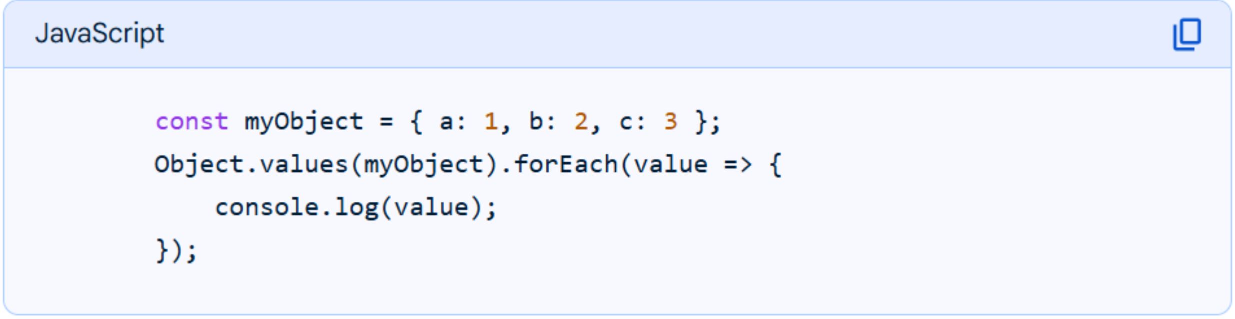
- `Object.keys()`:

    - Returns an array containing all the enumerable string-keyed property names of an object.
    - This array can then be iterated using array methods like `forEach()`, `for...of`, or a traditional `for` loop.
    - Example:



```javascript
const myObject = { a: 1, b: 2, c: 3 };
Object.keys(myObject).forEach(key => {
    console.log(`${key}: ${myObject[key]}`);
});
```

- `Object.values()`:
    - Returns an array containing all the enumerable property values of an object.
    - Useful when only the values are needed for iteration.
    - Example:



```javascript
const myObject = { a: 1, b: 2, c: 3 };
Object.values(myObject).forEach(value => {
    console.log(value);
});
```

- `Object.entries()`:
    - Returns an array of arrays, where each inner array represents a `[key, value]` pair for each enumerable property of the object.
    - Provides a convenient way to access both keys and values simultaneously during iteration.
    - Example:

JavaScript

```javascript
const myObject = { a: 1, b: 2, c: 3 };
Object.entries(myObject).forEach(([key, value]) => {
    console.log(`${key}: ${value}`);
});
```