**objects and its internal representation in Javascript**

**Introduction:**

JavaScript's most vital data type, objects, serves as the cornerstone for modern JavaScript programming. Unlike primitive data types (Number, String, Boolean, null, undefined, and symbol) that store a single value each, objects are more complex, capable of holding combinations of primitive and reference data types.

**Understanding Objects:**

Objects in JavaScript are reference data types, storing a reference or pointer to the location in memory where the object resides. Loosely defined as an unordered collection of related data in the form of "key: value" pairs, objects can include properties (variables) and methods (functions).

For example, envision an object representing a student with properties like name, age, address, and methods like updateAddress and updateName.

**Working with Object Properties:**

JavaScript objects have properties associated with them, acting as variables attached to the object. These properties define the characteristics of the object and can be accessed using dot notation or bracket notation.

**Example:**

var myCar = new Object();

myCar.make = 'Ford';

myCar.model = 'Mustang';

myCar.year = 1969;

**Property Accessors and Dynamic Properties:**

Object properties can be accessed or set using bracket notation, making objects akin to associative arrays. This notation is especially useful when dealing with dynamically determined property names or non-identifier names.

**Example:**

var myObj = new Object();

myObj['type'] = 'Dot syntax';

myObj['date created'] = 'String with space';

Iterating Over Object Properties:

The for...in loop allows iterating over all enumerable properties of an object. A function like showProps can display the properties of an object, filtering out properties from the object's prototype chain.

**Example:**

function showProps(obj, objName) {

var result = ``;

for (var i in obj) {

if (obj.hasOwnProperty(i)) {

result += `${objName}.${i} = ${obj[i]}\n`;

}

}

return result;

}

**Conclusion:**

Mastering JavaScript objects is pivotal for building robust and scalable applications. Whether manipulating properties, iterating over objects, or employing various creation methods, a deep understanding of objects is essential for unleashing the full power of JavaScript.