**2. JavaScript Objects and Their Representation:**

JavaScript is designed on a simple object-based paradigm. An object is a composite data type that allows you to store and organize data efficiently. Unlike primitive data types, objects can hold collections of data as key-value pairs.

In other words, object is a collection of properties and a property is an association between a **name** (or key) and **value.** A property’s value can be function, in which case the property is known as a method.

**Creating Objects:**

There are several ways to create objects in JavaScript:

1. **Object Literal:**

Declare a variable and assign value for that variable using inside the curly braces **{}** or using keyword new Object ().

For example,

const person = {

name: "Naveen",

age: 30,

isStudent: true

};

1. **Constructor Function:**

Define the object type writing a constructor function. There is a strong convention, with good reason, to use a capital initial letter.

For example,

function Person(name, age, isStudent) {

this.name = name;

this.age = age;

this.isStudent = isStudent;

}

const person = new Person("Naveen", 30, true);

1. **Class Syntax:**

Using class key word, we declare the object and inside the class using constructor key word declare the pair of keys and values.

For example,

class Person {

constructor(name, age, isStudent) {

this.name = name;

this.age = age;

this.isStudent = isStudent;

}

}

const person = new Person("Naveen", 30, true);

**Accessing Object Properties:**

You can access a property of an object by its property name. And accessors come in two syntaxes:

1. *Dot notation* (Objectname.key)
2. *Bracket notation* (Object[key)

An object property name can be any JavaScript string or symbol, including an empty string. However, you cannot use dot notation to access a property whose name is not a valid JavaScript identifier.

**Adding and Modifying Properties:**

Object in JS are mutable, which means you can add or modify properties at any time.

For example,

person.city = "New York"; // Adding

person.age = 31; // Modifying

**Object as Reference Types:**

JS objects are reference types, meaning that when you assign an object to a variable or pass it as a function argument, you’re working with a reference to the object memory. Changes made to the object are reflected in all references to tha object.

**Internal Representation of Objects:**

It essential to understand how JS engines internally represent objects for efficient storage and manipulation.

JS engines typically use hash tables or dictionaries to implement objects. These data structure allow fast access to values based on their keys. When you create an object and define properties, the engine stores them in this internal data structure.

In addition to, JS engines optimize objects for memory efficiency and execution speed. They use various techniques, such as hidden classes and inline caching, to make property access and modification as efficient as possible.