Object Literals, Constructors and OOP

What are Objects?

→ Objects can contain related code and data, representing information about the thing we are trying to model, and functionality that we want it to have. Object data is stored neatly inside an object package, making it easy to structure and access.

→ JavaScript is a powerful programming language that supports Object-Oriented Programming (OOP). In Javascript, objects created using object literal are singletons. This means when we made a change in an object, it affects the object entire the script. Whereas if an object is created using the constructor function, then the change will not affect the object throughout the script.

```
// constructor function.
FUNCTION MyCar() {};
// literal notation.
VAR MyCar = {};
```

→ Objects created using object literal:
→ Since, these are singletons, a change to an object affects the objects entire script. A change in one instance will affect all the instances of the object.

Object literal is a comma-seperated list of name-value pairs inside the curly braces. The object literals encapsulate the data. This minimizes the use of global variables, which can cause problems when combining the code.

Ex:

```
VAR MyCAR = {
    MAKE: 'FORD'
    MODEL: 'MUSTANG'
    YEAR: 2020
};
```

However, if we have to create multiple instances of a structure and perform operations based on predefined values, the we should use a function constructor

→ Objects created using the constructor function:

→ The object that is defined with a function constructor lets us have multiple instances of the object. If the change is made to one instance, it will not affect other instances. As we know that constructor is essentially a function that acts as a

Date

blueprint for creating objects. A convention for declaring constructors is always to capitalize its function name.

```
FUNCTION MyCar (MAKE, MODEL, YEAR) {
    THIS.MAKE = MAKE;
    THIS.MODEL = MODEL;
    THIS.YEAR = YEAR;
}
```

Now, we have our constructor function called MyCar. We can "construct" our object instances based on that blueprint.

```
// creating objects:
LET CAR1 = NEW MyCar ("FORD", "MUSTANG", 2000);
```

→ "NEW" keyword:

→ To create a new object instance, we use the "new" keyword to create an object based on a constructor. Adding the "NEW" keyword is a crucial step when creating an object from a constructor. The new keyword creates a new empty object. It binds property or function, which is declared with this keyword to the new object.