Prototype Inheritance

→ Inheritance :

→ Inheritance let the objects to share each other's properties. In other words, Inheritance is the process by which one object can be based on another.

→ Prototype Inheritance :

→ When we try to access a property or any object's method, the JavaScript will first search on the other itself, and then it will search the object's prototype if not found. If, after checking both the object and its prototype still no match is found, JavaScript will check the linked object prototype and continue seaching until the end of the prototype chain is reached. Object.prototype is at the end of the prototype chain. All the objects inherit the properties and methods of Object. When we try to search Beyond the end of the chain, results in null.

There are different ways to create an object in Javascript.

→ Object literal.
→ Function constructor.
→ The create() method.

→ OBJECT.CREATE():

→ The object.create() method using an existing object as the prototype, creates a new object.

Ex:

```
CONST MYDETAILS = {
    ISHUMAN: TRUE,
    PRINTINFO: FUNCTION() {
    CONSOLE.LOG(`MY NAME IS ${THIS.NAME}
    AM I HUMAN? ${THIS.HUMAN}`);
}};
CONST MYSELF= OBJECT.CREATE(MYDETAIL);
MYSELF.NAME = 'HARRY'; // "NAME" is a
    property set on "me", but not on "person"
MYSELF.PRINTINFO();
// Expected output: "My name is Harry. Am I
    human? true"
```

→ To create inheritance b/w function constructors call the parent constructor using call or link the prototype of the child constructor

to the parent constructor prototype.

→ Using call to chain constructors for an object:
→ The call() allows the method or function belonging to one object to be assigned and called for another object. This method provides a new value of this to the method or function. With call(), we can write a method once and then inherit it in another object without rewriting the new object's method.
For Ex.A: the constructor for the Factory object is defined with two parameters: name and location. Other functions, Food, invoke Factory, passing this, name, and location. Factory initializes the property's name and location; the specialized functions define the category.

```
FUNCTION FACTORY (NAME, LOCATION) {
    THIS.NAME = NAME;
    THIS.LOCATION = LOCATION;
}

FUNCTION FOOD (NAME, LOCATION) {
    FACTORY.CALL (THIS, NAME, LOCATION);
    THIS.CATEGORY = 'FOOD';
}
CONST MYFOOD = NEW FOOD ('NESTLE', 'UK');
```