Date: 02-6-2020
Morning Session: 9 am – 11 PM
By ~ Sundeep Charan Ramkumar Today

Topics: ES5 Classes

Is OOP (Object Oriented Programing Language) available in javascript ?? Ans : yes but it is not native of OOP. it is prototype based inheritance

This:

Use "this" keyword to represent the current object

```
var human = {
    name: "Sundeep Charan Ramkumar",
    age: 21,
    profession: "Instructor",
    introduce: function () {
      return (
        "Hi there. I am " +
        this.name +
        " of age " +
        this.age +
        " working as a " +
        this.profession
      );
    }
  };
```

Use "this" keyword to point out the current child object inside a function constructor.

```
// Custom data type
function Person(name, age, profession) {
  this.name = name;
  this.age = age;
  this.profession = profession;
}

var sundeep = new Person("Sundeep", 21, "Instructor");
var bagesh = new Person("Bagesh", 23, "Student");

console.log(bagesh, sundeep);
```

Function constructor: creates and initializes a **new function object** when called as a function rather than as a constructor. Thus the function call Function(…) is equivalent to the object creation expression **new Function(…)** with the same arguments.

MCQ 1) : Is inheritance under OOP or FP
Ans : OOP

MCQ 2) : when does the “this”keyword reqire
Ans : to create an Object

Common shareable method

```
function Person(name, age, profession) {
    this.name = name;
    this.age = age;
    this.profession = profession;
  }
  Person.prototype.introduce = function () {
    return (
      "Hi there. I am " +
      this.name +
      " of age " +
      this.age +
      " working as a " +
      this.profession
    );
  };

  var sundeep = new Person("Sundeep", 21, "Instructor");
  var bagesh = new Person("Bagesh", 23, "Student");
  console.log(bagesh, sundeep);
```

Inheritance: means Ability to extend an object and create a new object extracting a value of it..

Prototypical Inheritance

```
function Car(cc, isFourWheelDrive) {
    this.cc = cc;
    this.isFourWheelDrive = isFourWheelDrive;
  }

  Car.prototype.drive = function () {
    return "I am driving " + this.model;
  };

  Car.prototype.honk = function () {
    return "I am honking my";
  };

  Car.prototype.refuel = function () {
    return "I am refueling";
  };

  function Audi(model, color, cc, isFourWheelDrive) {
    var audi = new Car(cc, isFourWheelDrive);
    audi.model = model;
    audi.color = color;
    return audi;
  }

  // Extended the Car's function constructor to Audi's function constructor
  // Child prototpye = Parent prototype
  Audi.prototype = Car.prototype;

  Audi.prototype.drl = function () {
    return "I am blowinng up the DRL" + this.cc;
  };

  Audi.prototype.climateControl = function () {
    return "Climate control on";
  };

  var audiA8 = Audi("A8", "red", 3000, false);
  console.log(audiA8);
```

Resource:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Inheritance_and_the_prototype_chain