Assignment 6

Q1) Constructor: is used to create and initialize objects created from a class. It is the blueprint to create multiple objects(instances) that share same structure.

Its purpose is to set initial state or initialize the object and also set the properties and methods. Call constructor function uses “new” keyword.

Ex:

let Car = function (color, model) {

  //properties

  this.color = color;

  this.model = model;

};

let result = new Car("black", 2004);

console.log(result);

Q2) “this” keyword is used to refer the object. It allows us to use the properties and methods inside the object. Ex:

let person = {

  firstName: "Harvey",

  lastName: "Specter",

  city: "New York",

  birthYear: 1996,

  Education: "Laywer",

  getSummary: function () {

    return `${this.firstName} lived in ${this.city} and his birthYear is ${this.birthYear}`;

  },

};

console.log(person.getSummary());

Q3) call() : allows us to invoke a function and specify the value of “this”. It takes the ‘this’ value as first argument and after the comma the function argument.

Ex:

let mainPlane = {

  airline: "Fly India",

  iatacode: "FI",

  bookings: [],

  book: function (flightNum, name) {

    console.log(

      `${name} booked flight on ${this.airline} with fligh number: ${this.iatacode}${flightNum}`

    );

    this.bookings.push({

      flight: `${this.airline}`,

      name: name,

      flightNum: `${this.iatacode}${flightNum}`,

    });

  },

};

let childPlane = {

  airline: " childPlane",

  iatacode: "CP",

  bookings: [],

};

let book = mainPlane.book;

//book(233,"Louis") // book is regular fun and this value is undefined at least in strict mode

//solution : using "call" method

book.call(childPlane, 2423, "Mike");

mainPlane.book(552, "Carlos");

console.log(childPlane);

apply(): it is similar to call but here it takes ‘this’ object as the first argument and then array like object as the second argument.

Ex: //apply

book.apply(childPlane, [2424, "John"]);

The main difference between call() and apply() is in how the arguments are passed: apply() expects an array or an array-like object, while call() expects individual arguments. Both methods serve a similar purpose of manipulating the execution context and invoking functions with specific this values.

Bind(): it creates a new function with the same body as the original function but permanently sets the value of ‘this’ to a specific object.

Ex:

//bind

function greet() {

  console.log(`Welcome ${this.firstName} ${this.lastName} on my github`);

}

let user = {

  firstName: "Louis",

  lastName: "Litt",

};

let greets = greet.bind(user);

greets();

The main difference is that apply() immediately invokes the function, while bind() returns a new function with the desired context and arguments, allowing you to invoke it later.

Q5) Abstraction: Abstraction refers to hiding the implementation details of a code and exposing only the necessary information to the user. It provides the ability to simplify complex systems by ignoring irrelevant details and reducing complexity. Java provides many in-built abstractions and few tools to create our own.

Q6) Polymorphism: The word polymorphism can be broken down into Poly and morphs, as “Poly” means many and “Morphs” means forms. In simple words, we can say that ability of a message to be represented in many forms. The definition of polymorphism can be explained as performing a single task in different ways. A single interface having multiple implementations is also called polymorphism.

Q7) Inheritance: Inheritance is a mechanism wherein one class inherits the property of another. In inheritance, one class can adopt the methods and behavior of another class. It is a useful practice if you want to avoid writing the same piece of code repeatedly.

Q8) Encapsulation: it is a fundamental concept of Object-Oriented Programming, binds or bundles the related code units together, making the code more organized. Encapsulation helps to hide data by preventing unauthorized access to the implementation details.

Data Encapsulation can be defined as wrapping the code or methods(properties) and the related fields or variables together as a single unit. In object-oriented programming, we call this single unit - a class, interface, etc. We can visualize it like a medical capsule (as the name suggests, too), wherein the enclosed medicine can be compared to fields and methods of a class.

Q9) In JS, a class is a template or blueprint that defines the properties and behaviours of objects. It is a fundamental concept in OOP and provides a way to create objects with shared characteristics and methods.

To define a class in javascript, you use the ‘class’ keyword followed by the name of the class.

Q10) In JS, the ‘super’ keyword is used to call function or access the properties of an object’s parent. It is often used within a subclass to refer to the parent class and invoke its methods.

By using ‘super’ keyword, you can maintain the inheritance hierarchy and use the functionalities provided by the parent class while extending or modifying them in the subclass.