**Assignment 6 Solutions**

**Q.1 What’s Constructor And Its Purpose?  
Ans.** The constructor method is a special method of a class for creating and initializing an object instance of that class. A constructor enables you to provide any custom initialization that must be done before any other methods can be called on an instantiated object. If we don't provide any constructor, then a default constructor is provided.

**Q.2 Explain This Keyword and Its Purpose?  
Ans.** The **this** keyword is a reference variable that refers to the object that is currently executing the code. The reference of this keyword changes depending on how it is used.  
In an object method, **this** refers to the **object**.  
Alone, **this** refers to the global **object**.  
In a function, **this** refers to the **global object**.  
In a function, in strict mode, **this** is **undefined**.  
In an event, **this** refers to the **element** that received the event.  
Methods like call(), apply(), and bind() can refer **this** to **any object**.

Inside object this refers to the objectconst person = {  
    fName: "Manash",  
    lName: "Chauhan",

    displayName: function () {

        console.log(this);

        console.log(this.fName + " " + this.lName);

    }

}

person.displayName();

Output -   
{fName: 'Manash', lName: 'Chauhan', displayName: ƒ}

Manash Chauhan

Normally this refers to window  
console.log(this);

Window {window: Window, self: Window, document: document, name: '', location: Location, …}

const person = {

    fName: "Manash",

    lName: "Chauhan",

    displayName: function () {

        console.log(this);

        console.log(this.fName + " " + this.lName);

    }

}

person.displayName();

const person2 = {

    fName: "Ayushi",

    lName: "Chauhan",

    displayName: () => {

        console.log(this);

        console.log(this.fName + this.lName);

    }

}

person.displayName.call(person2);

{fName: 'Ayushi', lName: 'Chauhan', displayName: ƒ}

Ayushi Chauhan

**Q.3 What’s Call Apply Bind Method & Difference Between them.  
Ans. call** method is used to call an object method belonging to other object from a different object.  
Example: -

const myObj = {

firstName: 'Manash',

lastName: 'Chauhan',

displayName: function(calledFrom) {

console.log(`Hi my name is ${this.firstName} ${this.lastName}, I am called from ${calledFrom}`);

}

}

myObj.displayName('obj')

const myObj2 = {

firstName: "Object",

lastName: '2'

}

myObj.displayName.call(myObj2,'obj2');

Output: -

Hi my name is Manash Chauhan, I am called from obj

Hi my name is Object 2, I am called from obj2

**apply: -** Apply method is same as call except the way of taking arguments. Call takes each argument separately whereas apply takes single argument as an array.

const myObj = {

firstName: 'Manash',

lastName: 'Chauhan',

displayName: function(calledFrom, place) {

console.log(`Hi my name is ${this.firstName} ${this.lastName}, I am called from ${calledFrom} ${place}`);

}

}

myObj.displayName('obj', 'UK')

const myObj2 = {

firstName: "Object",

lastName: '2'

}

myObj.displayName.apply(myObj2,['obj2', "UP"]);

Output:

Hi my name is Manash Chauhan, I am called from obj UK

Hi my name is Object 2, I am called from obj2 UP

**bind: -**  using bind method, an object can borrow a method from a different object. Bind method does not invoke the method it rather returns it and we can invoke it later whenever we want.

const myObj = {

firstName: 'Manash',

lastName: 'Chauhan',

displayName: function(calledFrom, place) {

console.log(`Hi my name is ${this.firstName} ${this.lastName}, I am called from ${calledFrom} ${place}`);

}

}

*//myObj.displayName('obj', 'UK')*

const myObj2 = {

firstName: "Object",

lastName: '2'

}

const bindEg = myObj.displayName.bind(myObj2,['obj2', "UP"]);

bindEg();

Output: -  
Hi my name is Object 2, I am called from obj2,UP undefined

**Q.4 Explain OOPS ?  
Ans.** OOP is a programming paradigm that believes in grouping data (properties) and methods (actions) together inside a box. It demonstrates the pattern of real-world objects. Object-oriented programming is about modeling a system as a collection of objects, where each object represents some particular aspect of the system. Objects contain both functions (or methods) and data. An object provides a public interface to other code that wants to use it but maintains its own private, internal state; other parts of the system don't have to care about what is going on inside the object.

**Q.5 What is Abstraction and Its Purpose?  
Ans.** Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

**Q.6 Whats Polymorphism and Purpose of it?  
Ans.** The polymorphism is a core concept of an object-oriented paradigm that provides a way to perform a single action in different forms. It provides an ability to call the same method on different JavaScript objects. As JavaScript is not a type-safe language, we can pass any type of data members with the methods.  
Example: ‘+’ operator when passed with integer values will return their sum whereas when passed with strings will return concatenated string.

**Q.7 Whats Inheritance and Purpose of it?  
Ans.** Inheritance in JavaScript is defined as the ability of a class to derive properties and characteristics from another class while having its own properties as well.   
Inheritance helps to organize the data in a hierarchal form.  
It allows us to inherit properties from another class, thus making the class definitions less complex.  
Inheritance allows us to add our own properties in child classes as well; thus, we can override some methods of parent classes while inheriting the ones that are needed.

**Q.8 What is Encapsulation and Purpose of it ?  
Ans.** Encapsulation is defined as “the packing of data and functions into one component”. Packing, which is also known as bundling, grouping and binding, simply means to bring together data and the methods which operate on data. The component can be a function, a class or an object.  
Packing enables “controlling access to that component”. When we have the data and related methods in a single unit, we can control how is it accessed outside the unit. This process is called Encapsulation.  
Encapsulation provides protection of Object against Illegal Access.

**Q.9 Explain Class in JavaScript?  
Ans.** Classes are a template for creating objects. They encapsulate data with code to work on that data. Classes in JS are built on prototypes but also have some syntax and semantics that are unique to classes.  
Syntax:  
**class className {  
 // class body  
}**

**Q.10 What’s Super Keyword & What it does?  
Ans.** The super keyword is used to call the constructor of its parent class to access the parent's properties and methods.  
**Syntax:  
super(arguments);  
super.parentMethod(arguments);**