Assignment No.6

Q.1 What’s Constructor And Its Purpose?

Ans :- In object-oriented programming, a constructor is a special method that is used to initialize an object of a class. It is called automatically when an object is created and is responsible for initializing the object's state or setting its initial values.
The purpose of a constructor is to ensure that the object being created starts in a valid and consistent state by initializing its instance variables or properties. It allows for the initialization of object-specific data and the allocation of any necessary resources.

Q.2 Explain This Keyword and Its Purpose?

Ans :- In JavaScript, this keyword refers to the context in which a function is executed. Its value depends on how and where the function is called. The purpose of this keyword is to provide a reference to the object that owns or invokes the current executing function.

Q.3 What’s Call Apply Bind Method & Difference Between them?

Ans :- In JavaScript, the `call()`, `apply()`, and `bind()` methods are used to manipulate the `this` value and invoke functions with a specific context. Here's an explanation of each method and the differences between them:

1. `call()`: The `call()` method is used to invoke a function with a specified `this` value and arguments provided individually. It takes the `this` value as its first argument and subsequent arguments are passed directly to the function being called. The syntax for `call()` is: `functionName.call(thisValue, arg1, arg2, ...)`. This method allows you to pass arguments one by one.

2. `apply()`: The `apply()` method is similar to `call()`, but it accepts arguments as an array or an array-like object instead of individually. It is used to invoke a function with a specific `this` value and an array of arguments. The syntax for `apply()` is: `functionName.apply(thisValue, [arg1, arg2, ...])`. This method is useful when you have an array of arguments or when the number of arguments is not known beforehand.

The main difference between `call()` and `apply()` is how the arguments are passed: `call()` accepts arguments individually, while `apply()` accepts arguments as an array.

3. `bind()`: The `bind()` method is used to create a new function that has a specific `this` value and, optionally, preset arguments. Unlike `call()` and `apply()`, `bind()` does not immediately invoke the function. Instead, it returns a new function with the specified context and arguments. The syntax for `bind()` is: `functionName.bind(thisValue, arg1, arg2, ...)`. This method is useful when you want to create a new function with a fixed context or pre-set arguments that can be invoked later.

The main difference between `bind()` and the other two methods is that `bind()` returns a new function without invoking it, while `call()` and `apply()` immediately invoke the function with the specified context.
In summary, `call()` and `apply()` are used to invoke a function with a specific `this` value and arguments, while `bind()` is used to create a new function with a specific `this` value and optional arguments.

Q.4 Explain OOPS ?

Ans :- OOPS stands for Object-Oriented Programming System. It is a programming paradigm that organizes data and behavior into reusable structures called objects. OOPS revolves around the concept of objects, classes, inheritance, and encapsulation.

Q.5 What's Abstraction and Its Purpose?

Ans :- Abstraction is a fundamental concept in object-oriented programming (OOP) that focuses on representing the essential features of an object while hiding the unnecessary details. It provides a simplified and high-level view of an object or system, allowing users to interact with it without needing to understand the underlying complexities.
The purpose of abstraction is to:

Simplify Complexity: Abstraction helps manage complexity by breaking down a system into manageable modules. It allows developers to focus on the essential aspects of an object or system, while hiding the implementation details that are not relevant to the user.

Provide a Clear Interface: Abstraction defines a clear and well-defined interface that exposes only the necessary methods and properties to interact with an object. It hides the internal details and complexities, providing a simpler and more intuitive way to use the object.

Encapsulate Implementation: Abstraction encapsulates the implementation details of an object, making it easier to modify or update the underlying code without affecting the external usage. It promotes code maintainability and reduces the impact of changes on other parts of the system.

Enhance Code Reusability: By providing a high-level and generic interface, abstraction allows for code reusability. Objects and modules can be designed to work with different implementations, promoting modular and flexible software design.

Facilitate Team Collaboration: Abstraction provides a common understanding and communication platform between developers, enabling them to work collaboratively on complex projects. It helps in dividing the work, assigning responsibilities, and integrating different components seamlessly.

Q.6 What's Polymorphism and Purpose of it?

Ans :- Polymorphism is a concept in object-oriented programming (OOP) that allows objects of different classes to be treated as objects of a common superclass. It enables a single interface or method to have multiple implementations based on the specific objects involved. Polymorphism provides flexibility and extensibility to the code, allowing it to handle different object types seamlessly.

Q.7 What's Inheritance and Purpose of it?

Ans :- Inheritance is a fundamental concept in object-oriented programming (OOP) that allows a class to inherit properties and behaviors from another class. It establishes a hierarchical relationship between classes, where one class (called the subclass or derived class) inherits characteristics from another class (called the superclass or base class).

The purpose of inheritance is to promote code reuse, enhance code organization, and enable the creation of specialized classes based on existing ones

Q.8 What's Encapsulation and Purpose of it ?

Ans :- Encapsulation is a fundamental concept in object-oriented programming (OOP) that combines data and the methods that operate on that data into a single unit called a class. It involves wrapping data and methods together and controlling their access through a well-defined interface. The purpose of encapsulation is to ensure data integrity, enhance code maintainability, and promote information hiding.

Q.9 Explain Class in JavaScript?

Ans :- In JavaScript, a class is a blueprint or a template that defines the structure and behavior of objects. It serves as a way to create objects with specific properties and methods. Introduced in ECMAScript 2015 (ES6), the class syntax in JavaScript is syntactic sugar over the existing prototype-based inheritance model. It provides a more familiar and intuitive way to define classes and create objects.

Q.10 What’s Super Keyword & What it does?

Ans :- In JavaScript, the super keyword is used to call methods or access properties of a parent class within a subclass. It is primarily used in the context of inheritance, where a subclass extends a parent class and wants to utilize or override its methods or properties.
The super keyword in JavaScript provides a way to access and call the parent class constructor and methods within a subclass. It enables inheritance and allows subclasses to extend or override the functionality of the parent class.