Assignment Questions

1. A constructor is a special method in a class that is used for creating and

initializing objects of that class. It is called automatically when an object is created using the `new` keyword. The purpose of a constructor is to set initial values for the object's properties and perform any necessary setup or initialization tasks.

1. The `this` keyword in JavaScript refers to the current execution context or

the object that the function is being called on. Its value depends on how a function is invoked. The purpose of the `this` keyword is to provide a reference to the object that owns the currently executing code. It allows access to the object's properties and methods within the function.

1. The `call`, `apply`, and `bind` methods in JavaScript are used to set the value

of `this` and invoke a function in a specific context:

* + 1. The `call` method is used to invoke a function with a specified `this` value and arguments provided individually.
    2. The `apply` method is similar to `call`, but it takes an array-like object or an iterable as the second argument, which represents the arguments to be passed to the function.
    3. The `bind` method creates a new function with the same body as the original function but a different `this` value. It allows you to bind a specific object as the `this` value for the function.

The main difference between `call` and `apply` is in how the arguments are passed: `call` accepts arguments individually, while `apply` accepts arguments as an array or iterable. The `bind` method returns a new function with the specified `this` value bound to it, but it doesn't immediately invoke the function.

1. OOPS (Object-Oriented Programming) is a programming paradigm that organizes code around objects and their interactions. It focuses on creating reusable, modular, and scalable code by modeling real-world entities as objects. The key principles of OOPS are:
   1. Encapsulation: Encapsulation refers to bundling data and methods that operate on that data into a single unit called an object. It hides the internal details of an object and provides an interface to interact with it.
   2. Abstraction: Abstraction involves simplifying complex systems by breaking them down into smaller, more manageable units. It allows you to focus on essential features and hide unnecessary details.
   3. Inheritance: Inheritance allows objects to inherit properties and behaviors from other objects. It promotes code reuse and allows for the creation of hierarchical relationships between classes.
   4. Polymorphism: Polymorphism allows objects of different types to be treated as objects of a common superclass. It enables the use of a single interface to represent different types and provides flexibility and extensibility in code.
2. Abstraction is a fundamental concept in object-oriented programming that allows you to represent complex systems by focusing on the essential features and hiding unnecessary details. It involves creating abstract classes or interfaces that define a common set of methods or properties that concrete classes must implement.The purpose of abstraction is to simplify the usage and understanding of complex systems by providing a clear and concise interface. It allows you to work at a higher level of abstraction, focusing on the behavior and functionality rather than the implementation details. Abstraction promotes code reusability, modularity, and maintainability.
3. Polymorphism is the ability of an object to take on many forms or have multiple behaviors based on its data type or class hierarchy. It allows objects of different types to be treated as objects of a common superclass. Polymorphism is achieved through inheritance and method overriding. The purpose of polymorphism is to provide flexibility and extensibility in code. It allows you to write code that can work with objects of different types without explicitly knowing their specific implementations. Polymorphism enables code reuse, promotes modularity, and simplifies the addition of new types or classes without modifying existing code.
4. Inheritance is a key concept in object-oriented programming that allows objects to acquire properties and behaviors from other objects. It provides a way to create hierarchical relationships between classes, where a subclass (derived class) inherits the properties and methods of a superclass (base class). The purpose of inheritance is to promote code reuse and create a hierarchy of classes that share common characteristics. It allows for the creation of more specialized classes based on existing classes, reducing code duplication and improving maintainability. Inheritance enables the concept of "is-a" relationship, where a subclass is a specialized version of its superclass.
5. Encapsulation is a principle in object-oriented programming that involves bundling data and methods that operate on that data into a single unit called an object. It allows for the hiding of internal details and provides an interface to interact with the object. The purpose of encapsulation is to protect the integrity of data and ensure that it is accessed and modified only through controlled methods. It provides data abstraction and allows for the implementation details of an object to be hidden from the external world. Encapsulation enhances code maintainability, reusability, and security by preventing unauthorized access and accidental modifications to the internal state of an object.
6. In JavaScript, a class is a blueprint for creating objects that share similar properties and behaviors. It is a template or a constructor function that defines the structure and behavior of objects. Classes are introduced in ECMAScript 2015 (ES6) as syntactic sugar over JavaScript's existing prototype-based inheritance. A class in JavaScript consists of a constructor method and one or more prototype methods. The constructor method is used to initialize the object's properties when an object is created from the class. The prototype methods define the shared behavior of objects created from the class. Classes in JavaScript provide a more structured and familiar syntax for creating objects and defining their behaviors. They promote code organization, reusability, and support the principles of object-oriented programming.
7. The `super` keyword in JavaScript is used to call functions or access properties of a superclass (parent class) from within a subclass (child class). It is used in the context of inheritance to refer to the superclass's constructor or methods. When used in a constructor, `super()` is used to call the constructor of the superclass. This is necessary to initialize the inherited properties and set up the object's state properly. When used in methods, `super` allows you to access and invoke the methods of the superclass. It is useful when you want to override a method in the subclass but still want to use the implementation from the superclass. The `super` keyword helps in achieving code reuse, maintaining the inheritance chain, and extending the functionality of the superclass in the subclass.