Prototypal Inheritance Vs Classical Inheritance In JavaScript

In object Oriented Programming [OOP] Inheritance is a fundamental concepts that allows one class or object to Inherit properties and behaviors from another This mechanism Simplifies code reuse and allows objects to share attribute and method. while classical inheritance is widely used in many programming languages, prototypal inheritances is the model employed by JavaScript

i) Difference b/w Prototypal Inheritance and classical Inheritance

In classical Inheritance classes are defined as blueprints for creating objects. A class can inherits from another class using the extends keyword. Inheritance occurs by creating a class - based hierarchy where a child class inherits the properties and method of its parent class. This approach is common in language like Java, C++ and python.

On the other hand, prototypal inheritance is a model in which object can inherit directly from other objects. JavaScript, unlike classical languages, doesn't have the concept of classes in its core structure Instead each object has an internal property called a proto type if a property or method is not found in the object; Java Script will look for it in the prototype chain effectively allowing inheritance b/w objects.

2) How JavaScript Implements Inheritance Using Prototypes

In JavaScript all objects have a proto type object, which acts as a fall back object. If you try to access a property or method on an object and that property does not exist on the object itself JavaScript will look for it in the object prototype. if it's not found there it will search in the prototype's prototype and so on until it either finds the property or reaches the end of the prototype chain (null)

This is different from class-based inheritance where a Substance inherits from a class, and the inheritance chain is strictly determined by the class structure. In JavaScript you can add method directly to an object prototypes and any new object created from that prototype will automatically have access to those methods.

```
function Animal (name) {
    this.name = name;
}

Animal.prototype.sayHello = function ()
{
    console.log('${this.name} says
        hello!');
};

const dog = new Animal ('Dog');
  dog.sayHello();
const cat = new Animal ('cat');
  cat.sayHello();
```

Difference b/w prototype - Based and class - Based - inheritance

class Based	Prototype Based
• relies on a hierarchical structure of classes and subclasses	allows object to inherit directly from other object

x

The structure of the inheritance is defined when the class is declared	object can be dynamically modified at runtime.
Subclasses are explicitly defined	objects prototype is simply another object

Advantages

- Flexibility and Dynamism
- memory efficiency
- Simplicity and Intuition

Dis Advantages

- Prototype chain Lookup
- Potential for Inconsistent Design
- Lacks of Encapsulation

challenges

complexity with Deep prototype chains
In consistent Behaviour
Lack of Built in Structure