Prototype In Javascript

In JavaScript, the term "prototype" refers to a fundamental mechanism for implementing inheritance and sharing properties and methods among objects.

Key Concepts:

- **Prototype Property:**

  Every object in JavaScript has an internal property (often referred to as `[[Prototype]]` or accessed via `__proto__` in browsers, though `Object.getPrototypeOf()` is the standard way) that points to its prototype object.

- **Prototype Chain:**

  When you try to access a property or method on an object, JavaScript first checks if the property exists directly on that object. If not, it then looks for the property on the object's prototype. If still not found, it continues up the "prototype chain" by looking at the prototype's prototype, and so on, until either the property is found or the end of the chain (which is `null`) is reached.

- **Inheritance:**

  Prototypes enable inheritance in JavaScript. Objects can inherit properties and methods from their prototypes, promoting code reuse and reducing redundancy. When you create a new object, you can specify its prototype, effectively establishing an inheritance relationship.

- `prototype` Property of Functions:
  Functions in JavaScript also have a `prototype` property. When a function is used as a constructor (with the `new` keyword), the `prototype` property of that constructor function becomes the prototype for the newly created instances. This allows you to add shared methods and properties to instances created by that constructor.

  Example:

JavaScript

```javascript
// Constructor function
function Person(name) {
  this.name = name;
}

// Add a method to the Person's prototype
Person.prototype.greet = function() {
  console.log(`Hello, my name is ${this.name}.`);
};

// Create instances
const person1 = new Person("Alice");
const person2 = new Person("Bob");

// Call the inherited method
person1.greet(); // Output: Hello, my name is Alice.
person2.greet(); // Output: Hello, my name is Bob.
```