In JavaScript, prototypes and prototypal inheritance are key concepts that underlie the object-oriented nature of the language.
Here's a detailed explanation of each:

Prototype

A prototype is a mechanism by which JavaScript objects inherit properties and methods from other objects.
Every JavaScript object has a prototype, which is also an object.
When you create a new object, it inherits properties and methods from its prototype.

Prototype Chain

When you try to access a property on an object, JavaScript first looks at the object itself. If the property is not found, it then looks at the object's prototype, and then at the prototype's prototype, and so on, until it either finds the property or reaches the end of the prototype chain (usually Object.prototype).

```
const obj = {};
console.log(obj.toString);  // Output: function toString() { [native code] }
```

In this case, obj does not have a toString method directly. JavaScript looks up the prototype chain and finds toString in Object.prototype.
Similarly, we have Array.prototype, and in its prototype (which is an object), we have array methods like push, pop, etc.

Prototypal Inheritance

Prototypal inheritance is a feature in JavaScript where objects inherit properties and methods from other objects. Instead of classical inheritance (where classes inherit from other classes), JavaScript uses objects as the primary means of inheritance.

Example 1:

```
const person = {
    sayHello: function() {
        console.log(`Hello, my name is ${this.name}`);
    }
};

const alice = Object.create(person); // Creating alice object, which inherits from pers
alice.name = 'Alice';
alice.sayHello();  // Output: Hello, my name is Alice
```

Here, `person` is an object with a `sayHello` method. `alice` is created using `Object.create(person)`, so `alice` inherits from `person`.

Example 2:

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

Person.prototype.sayHello = function() {
    console.log(`Hello, my name is ${this.name}`);
};

const alice = new Person('Alice');
alice.sayHello();  // Output: Hello, my name is Alice
```

Difference Between `__proto__` and `prototype` {#difference-between-proto-and-prototype }

1. `__proto__` is used on object instances.
2. `prototype` is used on constructor functions.

Example of `__proto__` on Object Instances: {#example-of-proto-on-object-instances }

```
const obj = {};
obj.__proto__.sayHello = function() { console.log('Hello!'); };
obj.sayHello();  // Output: Hello!
console.log(obj);
```

Summary

1. A prototype is a mechanism by which JavaScript objects inherit properties and methods from other objects.
2. JavaScript provides built-in prototypes like `Number.prototype`, `String.prototype`, `Array.prototype`, and `Object.prototype`, each containing useful methods.
3. **Prototype Chain** - Every object has a prototype, forming a chain until reaching `null`. `Object.prototype.__proto__` is `null`.
4. **Prototypal Inheritance** - JavaScript objects inherit properties and methods from other objects rather than classes.
5. For every object created, Memory will be allocated only once for the methods and properties added to prototype