Prototypes and prototype chain

🔗 Prototypes and Prototype Chain in JavaScript – Explained in Detail

JavaScript is a **prototype-based language**, which means objects can inherit properties and methods from other objects using a mechanism called the **prototype chain**.

✅ What is a Prototype?

In JavaScript, every object has an internal link to another object called its **prototype**.

You can access an object's prototype using:

- `Object.getPrototypeOf(obj)`
- Or, internally via `__proto__` (though discouraged)

Example:

```
const person = {
  greet() {
    console.log("Hello");
  }
};

const student = Object.create(person); // Inherit from person
student.name = "Abhi";

student.greet(); // Hello
console.log(student.__proto__ === person); // true
```

✅ Prototype Chain

When you access a property or method on an object:

1. JavaScript checks if it exists **on the object itself**.
2. If not found, it looks **up the prototype chain**.

Visual example:

```
student → person → Object.prototype → null
```

If it finds the property along the chain, it returns it. Otherwise, it returns `undefined`.

✅ Functions and `prototype` Property

Every function in JavaScript has a `prototype` property that is used when the function is used as a **constructor** (with `new`).

```
function Car(make) {
  this.make = make;
}

Car.prototype.drive = function () {
  console.log(`${this.make} is driving`);
};

const car1 = new Car("Toyota");
car1.drive(); // Toyota is driving
```

Here:

- `car1.__proto__` === `Car.prototype`
- `Car.prototype.drive` is available to all instances created by `new Car()`

✅ `Object.prototype`

All standard objects inherit from `Object.prototype` unless explicitly created with `null`.

```
const obj = {};
console.log(obj.toString()); // [object Object]
console.log(obj.hasOwnProperty('x')); // false
```

These methods (`toString`, `hasOwnProperty`, etc.) are from `Object.prototype`.

✅ Checking the Prototype Chain

You can use:

```
object instanceof Constructor
```

Or:

```
Constructor.prototype.isPrototypeOf(object)
```

Example:

```
console.log(car1 instanceof Car); // true
console.log(Car.prototype.isPrototypeOf(car1)); // true
```

✅ Summary Table

Concept	Description
`prototype`	Property on constructor functions for inheritance
`__proto__`	Internal reference to the prototype object
`Object.create(obj)`	Creates new object with `obj` as its prototype
Prototype Chain	Lookup sequence: object → prototype → ... → null

---

🧠 Quick Analogy

Think of prototypes like a chain of folders:

- You look in the current folder (object),
- If it's not there, you look in the next (prototype),
- and so on until you reach the end (`null`).

Let me know if you'd like a visual diagram or quiz to reinforce this topic!