instanceof and isPrototypeOf

Let's explore two important operators used in JavaScript to check inheritance and prototype relationships: `instanceof` and `isPrototypeOf`.

🔍 1. `instanceof` Operator

✅ Purpose:

Checks whether an object is an **instance of a constructor or class**, including inheritance in the prototype chain.

🧠 Syntax:

```
object instanceof Constructor
```

🛠️ Example:

```
function Animal() {}
function Dog() {}

Dog.prototype = Object.create(Animal.prototype);
const myDog = new Dog();

console.log(myDog instanceof Dog);      // true
console.log(myDog instanceof Animal);  // true
console.log(myDog instanceof Object);  // true
```

💡 Notes:

- It checks **if Constructor.prototype exists somewhere in the object's prototype chain**.
- Works with **custom constructors**, **built-ins**, and **ES6 classes**.

🧬 2. `isPrototypeOf()` Method

✅ Purpose:

Checks whether an **object exists in another object's prototype chain**.

🧠 Syntax:

```
prototypeObj.isPrototypeOf(object)
```

🛠️ Example:

```
function Animal() {}
function Dog() {}

Dog.prototype = Object.create(Animal.prototype);
const myDog = new Dog();

console.log(Animal.prototype.isPrototypeOf(myDog)); // true
console.log(Dog.prototype.isPrototypeOf(myDog));    // true
console.log(Object.prototype.isPrototypeOf(myDog)); // true
```

💡 Notes:

- Useful when **working with raw prototypes**, especially in **manual inheritance setups**.
- Returns `true` if the calling object is found in the prototype chain of the given object.

🔁 Difference Between `instanceof` and `isPrototypeOf`

Feature	`instanceof`	`isPrototypeOf`
Checks against	Constructor function	Prototype object
Returns	`true` or `false`	`true` or `false`

Typical usage	`obj instanceof Constructor`	`Constructor.prototype.isPrototypeOf(obj)`
Use-case	Checking class instances	Inspecting prototype chain

👀 Bonus: Edge Case

`instanceof` can fail if an object is created across different realms (e.g., iframes), because the constructor won't match.

```
// cross-realm instanceof can be tricky and fail
```

For strict prototype inspection, `isPrototypeOf()` is **more reliable** in those edge cases.

Let me know if you want examples using ES6 classes, `Object.create`, or prototypes in complex chains.