**04 Prototypical Inheritance**

**1) Creating Your Own Prototypical Inheritance**:

Suppose we have a Circle object and in this circle object we have an instance property “radius” and two prototype property “draw()” and “duplicate()”

function Circle(radius) {

*this*.radius = radius;

}

Circle.prototype.draw = function() {

console.log("draw");

};

Circle.prototype.duplicate = function() {

console.log("duplicate");

};

Now we want to add a Square object here that Square object also have a duplicate method with the exact same implementation. We don’t want to repeat the implementation. We want to inheritance.

So, we define a “Shape” object and put this duplicate method here, and then have Square and Circle inherited from the Shape object.

**Example**:

function Shape() {}

Shape.prototype.duplicate = function() {

console.log("duplicate");

};

function Circle(radius) {

*this*.radius = radius;

}

Circle.prototype.draw = function() {

console.log("draw");

};

const s = new Shape();

const c = new Circle(20);

console.log(s);

*/\**

*Shape {}*

*\_\_proto\_\_:*

*duplicate: ƒ ()*

*constructor: ƒ Shape()*

*\_\_proto\_\_: Object //objectBase*

*\*/*

console.log(c);

*/\**

*Circle {radius: 20}*

*radius: 20*

*\_\_proto\_\_:*

*draw: ƒ ()*

*constructor: ƒ Circle(radius)*

*\_\_proto\_\_: Object //objectBase*

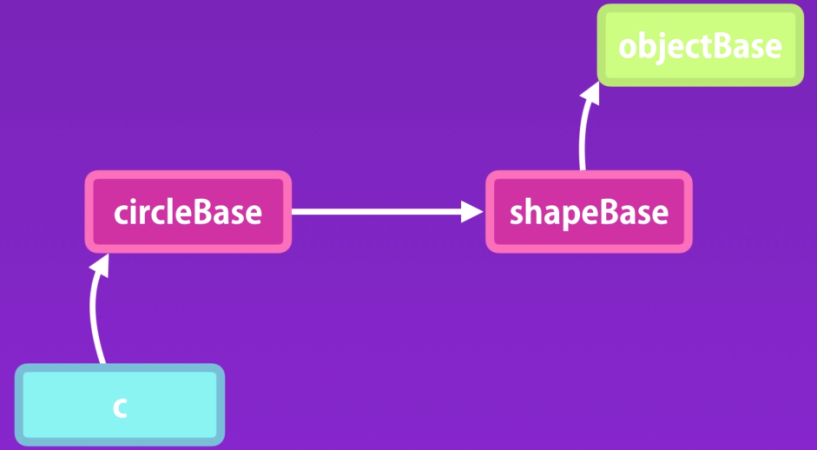
*\*/*

Here in memory

c (Object)-----(inherit from)----->circelBase(Circle.prototype)-----(inherit from)----->objectBase

s (Object)-----(inherit from)----->shapeBase(Shape.prototype)-----(inherit from)----->objectBase

Now to set up inheritance here, we want to have circleBase inherit from shapeBase.



**Example**:

function Shape() {}

Shape.prototype.duplicate = function() {

console.log("duplicate");

};

function Circle(radius) {

*this*.radius = radius;

}

*//circleBase is like this*

*//Circle.prototype = Object.create(Object.prototype); //objectBase*

*//Create ShapeBase prototype and set to to CircleBase*

Circle.prototype = Object.create(Shape.prototype);

Circle.prototype.draw = function() {

console.log("draw");

};

const s = new Shape();

const c = new Circle(20);

console.log(s);

*/\**

*Shape {}*

*\_\_proto\_\_:*

*duplicate: ƒ ()*

*constructor: ƒ Shape()*

*\_\_proto\_\_: Object*

*\*/*

console.log(c);

*/\**

*Circle {radius: 20}*

*radius: 20*

*\_\_proto\_\_: Shape*

*draw: ƒ ()*

*\_\_proto\_\_:*

*duplicate: ƒ ()*

*constructor: ƒ Shape()*

*\_\_proto\_\_: Object*

*\*/*

This is prototypical inheritance.

**2) Resetting the Constructor**:

Every object in JavaScript have a constructor property, that returns the function that was use to construct or create that object.

We can create object by using prototype constructor

**Example**:

function Shape() {}

Shape.prototype.duplicate = function() {

console.log("duplicate");

};

function Circle(radius) {

*this*.radius = radius;

}

Circle.prototype.draw = function() {

console.log("draw");

};

const s = new Shape();

const c = new Circle(20);

console.log(new Circle.prototype.constructor(20));

*/\**

*Circle {radius: 20}*

*radius: 20*

*\_\_proto\_\_:*

*draw: ƒ ()*

*constructor: ƒ Circle(radius)*

*\_\_proto\_\_: Object*

*\*/*

Here, "new Circle.prototype.constructor(20)" is equivalent to “new Circle(20)”

**Example**:

console.log(new Circle(20));

*/\**

*Circle {radius: 20}*

*radius: 20*

*\_\_proto\_\_:*

*draw: ƒ ()*

*constructor: ƒ Circle(radius)*

*\_\_proto\_\_: Object*

*\*/*

But now we replace the circleBase with shapeBase.

Circle.prototype = Object.create(Shape.prototype);

Now the constructor property of Circle is not present in Circle object. It replaces with shapeBase. The constructor property is now the Shape() function.

**3) Calling the Super Constructor**:

**4) Intermediate Function Inheritance**:

**5) Method Overriding**:

**6) Polymorphism**:

**7) When to Use Inheritance**:

**8) Mixins**:

**1) Exercise Prototypical Inheritance**:

**1) Solution Prototypical Inheritance**:

**2) Exercise- Polymorphism**:

**2) Solution- Polymorphism**:

04 Prototypical Inheritance