Classes

Classes are syntactic sugar over prototypical inheritance.

Class Declaration class Circle

{

constructor(radius)

{

this.radius = radius; this.move = function()

{

console.log('move');

}

}

draw()

{

console.log('draw');

}

}

const circle = new Circle(1);

console.log(circle); //Circle { radius: 1, move: [Function (anonymous)] }

Class Expression

const Square = class{}

Static Methods

class Car

{

constructor(name)

{

this.name = name;

}

static hello()

{

return "Hello!!";

}

}

console.log(Car.hello()); // Hello!!

this Keyword

function Circle()

{

this.draw = function()

{

console.log(this);

}

}

const circle = new Circle(1);

circle.draw(); //Circle { draw: [Function (anonymous)] }

const draw = circle.draw;

draw(); // Window Object or global Object 'use strict';

function Circle()

{

this.draw = function()

{

console.log(this);

}

}

const circle = new Circle();

circle.draw(); //Circle { draw: [Function (anonymous)] }

const draw = circle.draw; draw(); // undefined

By default, class body is executed under strict mode.

class Circle

{

constructor()

{

this.draw = function()

{

console.log(this);

}

}

}

const circle = new Circle();

circle.draw(); //Circle { draw: [Function (anonymous)] }

const draw = circle.draw; draw(); //undefined

Private Members using Symbols

Method1: class Circle

{

constructor(radius)

{

this.\_radius = radius;

}

}

Method2:

const \_radius = Symbol(); const \_draw = Symbol();

class Circle

{

constructor(radius)

{

//we have to use bracket notataion instead of dot notation. this[\_radius] = radius;

}

[\_draw]()

{

console.log('draw');

}

}

const c = new Circle(1); console.log(Object.getOwnPropertyNames(c)); //[]

const keys = Object.getOwnPropertySymbols(c); console.log(keys[0]); //Symbol() console.log(c[keys[0]]); //1

Private Members using WeakMaps

const \_radius = new WeakMap(); const \_move = new WeakMap();

class Circle

{

constructor(radius)

{

\_radius.set(this, radius);

\_move.set(this, function()

{

console.log('move', this);

})

}

draw()

{

\_move.get(this)(); console.log(\_radius.get(this)); console.log('draw');

}

}

const c = new Circle(1); c.draw();

//move undefined

//1

//draw

const \_radius = new WeakMap(); const \_move = new WeakMap();

class Circle

{

constructor(radius)

{

\_radius.set(this, radius);

\_move.set(this, ()=>

{

console.log('move', this);

})

}

draw()

{

\_move.get(this)(); console.log(\_radius.get(this)); console.log('draw');

}

}

const c = new Circle(1); c.draw();

//move Circle {}

//1

//draw

Getters and Setters

const \_radius = new WeakMap();

class Circle

{

constructor(radius)

{

\_radius.set(this, radius);

}

get radius()

{

return \_radius.get(this);

}

set radius(value)

{

if(value < 0)

{

throw new Error("Enter positive radius");

}

\_radius.set(this, value);

}

}

const c = new Circle(1); console.log(c.radius); //1 c.radius = 5; console.log(c.radius); //5

c.radius = -5; //Error: Enter positive radius

Inheritance

class Shape

{

constructor(color)

{

this.color = color;

}

move()

{

console.log('move');

}

}

class Circle extends Shape

{

constructor(color, radius)

{

super(color); this.radius = radius;

}

draw()

{

console.log('move');

}

}

const c = new Circle();

console.log(Circle. proto ); //[class Shape] console.log(Circle. proto . proto ); //{}

Method Overriding

class Shape

{

move()

{

console.log('move');

}

}

class Circle extends Shape

{

move()

{

super.move(); console.log('Circle move');

}

}

const c = new Circle(); c.move();

//move

//Circle move