**05 ES6 Classes**

**1) ES6 Classes**:

JavaScript classes, introduced in ECMAScript 2015, are primarily syntactical sugar over JavaScript's existing prototype-based inheritance. The class syntax does not introduce a new object-oriented inheritance model to JavaScript.

Classes are in fact "special functions", and just as you can define function expressions and function declarations, the class syntax has two components: class expressions and class declarations.

**Example**:

*/\**

*function Circle(radius) {*

*this.radius = radius;*

*this.draw = function() {*

*console.log("draw");*

*};*

*}*

*\*/*

*//write code using ES-6 class*

class Circle {

*//instance property*

constructor(radius) {

*this*.radius = radius;

*this*.move = function() {

console.log("move");

};

}

*//prototypical property*

draw() {

console.log("draw");

}

}

const c = new Circle(10);

console.log(c); *//Circle {radius: 10, move: ƒ}*

*/\**

*Circle {radius: 10, move: ƒ}*

*move: ƒ ()*

*radius: 10*

*\_\_proto\_\_:*

*constructor: class Circle*

*draw: ƒ draw()*

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

*\*/*

In JavaScript classes are essentially function.

console.log(typeof Circle); *//function*

**2) Hoisting**:

Hoisting is JavaScript's default behavior of moving declarations to the top.

In JavaScript we can define a class in two ways by using declaration or by using expression syntax.

**Example**:

*//class Declaration*

class Circle{

}

*//class Expression*

const Square = Class{

}

Unlike functions class declaration or class expression are not hoisting. If we cannot create the Circle object like this.

**Example**:

const c = new Circle(); *//Uncaught ReferenceError: Circle is not defined*

class Circle{

}

**Note**:

The recommended approach to declare a class in JavaScript is "using declaration". Using Expression

is not used.

**3) Static Methods**:

The static keyword defines a static method for a class. Static methods aren't called on instances of the class. Instead, they're called on the class itself. These are often utility functions, such as functions to create or clone objects.

**Example**:

class Circle {

constructor(radius) {

*this*.radius = radius;

}

draw() {

console.log("draw");

}

static possition() {

console.log("possition");

}

}

const c = new Circle(20);

console.log(c); *//Circle {radius: 20}*

*/\**

*Circle {radius: 20}*

*radius: 20*

*\_\_proto\_\_:*

*constructor: class Circle*

*draw: ƒ draw()*

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

*\*/*

The static member is not available in object level it is available only class level. We can access static member as following.

**Example**:

class Circle {

constructor(radius) {

*this*.radius = radius;

}

draw() {

console.log("draw");

}

static possition() {

console.log("possition");

}

}

console.log(Circle.possition()); *//position*

**4) The this keyword**:

The JavaScript this keyword refers to the object it belongs to.

1. In a method, this refers to the owner object.
2. Alone, this refers to the global object.
3. In a function, this refers to the global object.
4. In a function, in strict mode, this is undefined.
5. In an event, this refers to the element that received the event.
6. Methods like call(), and apply() can refer this to any object.

**Inside function this point to the current object**:

const Circle = function() {

*this*.draw = function() {

console.log(*this*);

};

};

const c = new Circle();

c.draw(); *//Circle {draw: ƒ}*

*/\**

*Circle {draw: ƒ}*

*draw: ƒ ()*

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

*\*/*

**Here this point to window object**:

const Circle = function() {

*this*.draw = function() {

console.log(*this*); *//this refer window now*

};

};

const c = new Circle();

const draw = c.draw;

*//function call*

draw();

*/\**

*Window {postMessage: ƒ, blur: ƒ, focus: ƒ, close: ƒ, parent: Window, …}*

*\*/*

Here we call "draw()" as a standalone function which is not part of the object. When we call draw() as a function "this" point to the window object in browser.

**In a function, in strict mode, this is undefined**:

"use strict";

const Circle = function() {

*this*.draw = function() {

console.log(*this*);

};

};

const c = new Circle();

const draw = c.draw;

*//function call*

draw(); *//undefined*

**By default, the body of our classes is strict mode ("use strict")**

class Circle {

draw() {

console.log(*this*);

}

}

const c = new Circle();

const draw = c.draw;

*//function call*

draw();

*//undefined*

**5)** **Private Members Using Symbols**:

**6) Private Members Using WeakMaps**:

**7) Getters and Setters**:

**8) Inheritance**:

**9) Static Methods**:

**10) Exercise**:

**11) Solution**:

05 ES6 Classes