**Super Keyword in JavaScript**

The super keyword in JavaScript is used to call the constructor of a parent class.

**Definition**

* **super**: A keyword used to call functions on an object's parent class.

#### **Calling Parent Constructor**

class Parent {

constructor(name) {

this.name = name;

}

}

class Child extends Parent {

constructor(name, age) {

super(name); // Calls Parent's constructor

this.age = age;

}

}

const child = new Child('John', 25);

console.log(child.name); // John

console.log(child.age); // 25

#### **2. Calling Parent Methods**

class Parent {

greet() {

return 'Hello from Parent';

}

}

class Child extends Parent {

greet() {

return `${super.greet()} and Hello from Child`;

}

}

const child = new Child();

console.log(child.greet()); // Hello from Parent and Hello from Child

**this Keyword in JavaScript**

The this keyword in JavaScript refers to the object that is executing the current function. Its value depends on the context in which it is used.

**Definition**

* **this**: A keyword that refers to the current execution context of a function.

**Reasons and Usage**

1. **Object Methods**:
   * In an object method, this refers to the object to which the method belongs.
2. **Constructor Functions and Classes**:
   * In a constructor function or a class, this refers to the instance of the object being created.
3. **Global Context**:
   * In the global execution context (outside of any function), this refers to the global object (e.g., window in browsers, global in Node.js).
4. **Event Handlers**:
   * In an event handler, this refers to the element that received the event.

#### **Object Methods**

const person = {

name: 'Alice',

greet: function() {

console.log(this.name); // 'this' refers to the 'person' object

}

};

person.greet(); // Alice

#### Constructor Functions and Classes

function Person(name) {

this.name = name;

}

const person1 = new Person('Bob');

console.log(person1.name); // Bob

class Car {

constructor(brand) {

this.brand = brand;

}

displayBrand() {

console.log(this.brand); // 'this' refers to the instance of the Car

}

}

const car1 = new Car('Toyota');

car1.displayBrand(); // Toyota

**this.state**: Initializes the component's state. The state is an object that holds data that can change over the lifecycle of the component.

constructor() {

super();

this.state = {

message: 'Hello Manisha',

};

}

**this.setState()**: This method is provided by React's Component class. It schedules an update to the component's state object. When the state changes, the component re-renders.

**render Method**:

* The render method is required in all class components. It returns the JSX (JavaScript XML) that defines what the UI of the component looks like.
* <h1>{this.state.message}</h1>: An <h1> element that displays the current value of this.state.message. The curly braces {} allow you to embed JavaScript expressions within JSX.

render() {

return(

<div>

<h1>{this.state.message}</h1>

<button onClick={() => { this.changeMessage() }}>Subscribe</button>

</div>

); }