Solution 1:

```
   function Person(name, age) {
    this.name = name;
    this.age = age;

    this.sayHello = function() {
     console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);
    };
}

// Example usage:
const person1 = new Person('Kiran', 30);
person1.sayHello();

const person2 = new Person('Mala', 25);
person2.sayHello();
```

Solution 2:

```
function Person(name, age) {
    this.name = name;
    this.age = age;
}

Person.prototype.sayHello = function() {
    console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);
};

function Employee(name, age, designation) {
    // Call the Person constructor
    Person.call(this, name, age);
    this.designation = designation;
}

Employee.prototype = Object.create(Person.prototype);
Employee.prototype.constructor = Employee;

Employee.prototype.getDetails = function() {
    console.log(`Name: ${this.name}, Age: ${this.age}, Designation: ${this.designation}`);
};

// Example usage:
const employee1 = new Employee('Alice', 30, 'Software Engineer');
employee1.sayHello();
employee1.getDetails();

const employee2 = new Employee('Bob', 25, 'Product Manager');
employee2.sayHello();
employee2.getDetails();
```

Solution 3:

```
                        function Calculator() {
                        this.value = 0;
                        }

                        Calculator.prototype.add = function(num) {
                            this.value += num;
                            return this; // Return the Calculator instance for chaining
                        };


                        Calculator.prototype.subtract = function(num) {
                          this.value -= num;
                          return this; // Return the Calculator instance for chaining
                        };


                        Calculator.prototype.multiply = function(num) {
                          this.value *= num;
                            return this; // Return the Calculator instance for chaining
                        };

                        Calculator.prototype.divide = function(num) {
                          if (num !== 0) {
                              this.value /= num;
                           }
                         else {
                                     console.error('Cannot divide by zero.');
                                  }
                return this;
            };

            Calculator.prototype.getValue = function() {
              return this.value;
            };

// Example usage:
            const calc = new Calculator();

            const result = calc.add(10)
                                     .subtract(2)
                                     .multiply(3)
                                     .divide(2)
                                     .getValue();

                                    console.log(result);
```

Solution 4:

```
class Shape {
draw() {
console.log('Drawing a shape.');
 }
}
class Circle extends Shape {
 draw() {
  console.log('Drawing a circle.');
}
}

class Rectangle extends Shape {
    draw() {
  console.log('Drawing a rectangle.');
}
}


function renderShape(shape) {
 shape.draw();
}

const circle = new Circle();
const rectangle = new Rectangle();


renderShape(circle);
renderShape(rectangle);
```

Solution 5:

```
            Array.prototype.customIncludes = function(element, fromIndex = 0) {
             fromIndex = Math.max(fromIndex, 0);

                         for (let i = fromIndex; i < this.length; i++) {
                            if (this[i] === element) {
                                    return true;
                                     }
                        }
                        return false;
                        };

// Example usage:
            const arr = [1, 2, 3, 4, 5];

console.log(arr.customIncludes(3));
console.log(arr.customIncludes(6));
console.log(arr.customIncludes(2, 2));
console.log(arr.customIncludes(3, 1));
```