In an arrow function the this points to the surrounding scope or the parent scope, enclosing scope

Classes in Javascript

1) Primitive way
2) Modern way

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

var dog = new Dog(‘barfi’, 2, ‘Indie’);
```

Create a pizza contructor function

toppings
size

crustType

describe method

Output
___size___ of the pizza with ___ topping on a ___ crust

```
function Pizza(toppings, size, crustType) {
    this.toppings = toppings;
    this.size = size;
    this.crustType = crustType;

    this.describe = function() {
        console.log(`A ${this.size} pizza with
${this.toppings.join(", ")} on a ${this.crustType} crust.`);
    };
}


var customerOrder1 = new Pizza(['cheese', 'pepperoni'],
'medium', 'thin');
```

```
class Pizza {

	constructor(toppings, size, crustType) {
		this.toppings = toppings;
		this.size = size;
		this.crustType = crustType;
	}

	describe() {
		console.log(`A ${this.size} pizza with
	${this.toppings.join(", ")} on a ${this.crustType} crust.`);

	}
}
```

Inheritance

StuffedCrustPizza

Cheese
Garlic
Mushroom
Sweet Onions

```
class Pizza {

    constructor(toppings, size, crustType) {
        this.toppings = toppings;
        this.size = size;
        this.crustType = crustType;
    }

    describe() {
        console.log(`A ${this.size} pizza with
    ${this.toppings.join(", ")} on a ${this.crustType} crust.`);

    }
}


Class StuffedCrustPizza extends Pizza {
    constructor(toppings, size, crustType, stuffingType) {
        super(toppings, size, crustType) // call the parent
class constructor with super
        this.stuffingType = stuffingType;
    }

    describeStuffing() {
        console.log(`This Pizza has ${this.stuffingType}
stuffing in the crust.`);
    }
    describe() {
        super.describe();
        this.describeStuffing();
    }


}
```

Static Property and Static Method

Static Methods belong to the class rather than any particular object instance. They can be called directly on the class itself.

```
class Pizza {

    static totalPizzasMade = 0;
    constructor(toppings, size, crustType) {
        this.toppings = toppings;
        this.size = size;
        this.crustType = crustType;
        Pizza.totalPizzasMade++;
    }

    describe() {
            console.log(`A ${this.size} pizza with
    ${this.toppings.join(", ")} on a ${this.crustType} crust.`);

    }

    static totalPizzasMade() {
     console.log(`Total pizzas made:
${Pizza.totalPizzasMade}`);

    }
}
```

```
class MathUtils {
  static add(a, b) {
    return a + b;
  }

  static multiply(a, b) {
    return a * b;
  }
}

console.log(MathUtils.add(2, 3));       // 5
console.log(MathUtils.multiply(4, 5));  // 20
```