Object Oriented In JavaScript.

Classes

classes are the blue print on a set of real life entities.
classes are going to represent that how a entity should look like and behave. When I say how do they look , I refer to the properties they posses. And when I say behave then i mean what actions can be performed on them.

for example `Product`
`aggar mein product ka example lunga toh us product mein kya kya properties aaenge.` . price
. `company`
. `reviews`
. `rating`
. `description`
so yeah sare hamare properties hai product department ke in otherwords ham usko Data members bhi bol sakhte hein`

Aur aggar mein beheviour ke bare mein baat karun toh us mein bahut kuchj hai jaise ki
.we can buy a product
. we can display details of a product.
. we can wishlist a product
. we can rate a product.
. we can add review to the product
. add product to the cart

`yeah toh ho gaya sara action ke bare mein eska ek technical term bhi hai jisko ham bolte hein Member Functions.`

How to make class in JS

We can create a class by using class keyword and inside the class we can write out Member Function and Data Members

for example how to create a class:

```
class NameOfTheClass {
        // data members
        // member functions
}
```

Member Functions

While creating `Member Functions` no need of writing function keyword , we can just create it by writing the `functionName` followed by a pair of parenthesis for example `addToCart`

`Member function matlab aap function hi bana rahe ho but function keyword ke bina`

```
class Product{

        addToCart(){
                console.log('Product added to the cart')
        }
        removeFromCart(){
                console.log('Product removed from the cart')
        }
        displayProduct(){
                console.log('Product Displayed')
        }
        buyProduct(){
                console.log('Product Brought')
        }


}
```

Data Member

While creating a Data Member no need of writing `let, const, var`

`Data Member matlab aap variable hi bana rahe ho but let, var, const ke bina`

```
class Product{

        name;
        price;
        category;
        description;
        rating;

}
```

Constructor : a very special member function of every class:

Every class that we make in JS, has one special member function called as **constructor**
and the constructor is automatically called whenever we will create an object `(jab bhi ham objects banaenge constructor automatically called ho jaega)`. The constructor is the first method to be called automatically when ever we will create an object with the help of classes.

The default version of this constructor is called default constructor and we can change the implementation of this constructor by writing our own. `

`( har class mein ek contsructor exist karta hai jisko ham bolte hein default constructor aur woh hamesha bydefault khudse hi call ho jata hein and uske andar kuch hota bhi nahi hai woh empty hota hai. )`

In JavaScript if you want to create a constructor just write constructor followed by parenthesis and curly braces.

```
class Product {
        constructor(){
                // this is our custom constructor and it will be called when a new
                // object will be created
        }
}
```

Why the constructor is diffrent from other because this is the only function which will be called automatically and there no other member functions which will be called automatically remember one thing that constructor is also called a Member Function

Objects:

Using classes the final entity we develop is called as objects.

How to create Objects in JavaScript.

JavaScript has a keyword called `new` by the help of which we can create an objects.

```
class Product{
        let name = name // this is not allowed here.
        name;
        price;
        category;
        description;
        rating;

        addToCart(){
                console.log('Product added to the cart')
        }
        removeFromCart(){
                console.log('Product removed from the cart')
        }
        displayProduct(){
                console.log('Product Displayed')
        }
        buyProduct(){
                console.log('Product Brought')
        }


}
```

```
let vivo = new Product() // object created
```

New Keyword

Let's talk what does this new keyword is doing internally. Every time we call new it does the following 4 step procedure.

1. it does creates a brand new empty object.
2. it calls the constructor of the class and passes the `newely` created object inside a keyword `this` . Constructor has automatically access to the this keyword and when we call `new` then the `this` keyword has access to the `newely` created object in step 1 and constructor now can use the this keyword inside it. And then whatever the logic of constructor gets executed ` [pehla step mein empty object banta hai and dusra step mein constructor ko call karenge and jo empty object pehle banaya hoga usko aap constructor ke andar jo this keyword ka access hota hein us mein empty object ko pass kar denge but not as parameter]
3. In step 3 `new` triggers everything needs to be done for prototype to work.
4. Now if from a constructor an object is manually returned then the manually returned object is stored in the called variable and we will get that object, otherwise in any other case i.e. if you don't return anything manually then we will get the Product. (returned the value inside the this keyword) `(aagar aap constructor mein kuch object return karte ho then woh object jake store ho jaega vivo variable mein and jab ham console.log(vivo) karange toh hame custom wala object milega jo hamne return kiya hoga constructor mein otherwise aggar hamne manually kuch return nahi kiya toh hame milega normal vivo ka Product ya phir object ke alawa aggar meine kuch aur return kiya constructor se toh hame wohi wala milega aggar meine return kiya [10,20] toh mujhe yahi wala array milega.

for example:

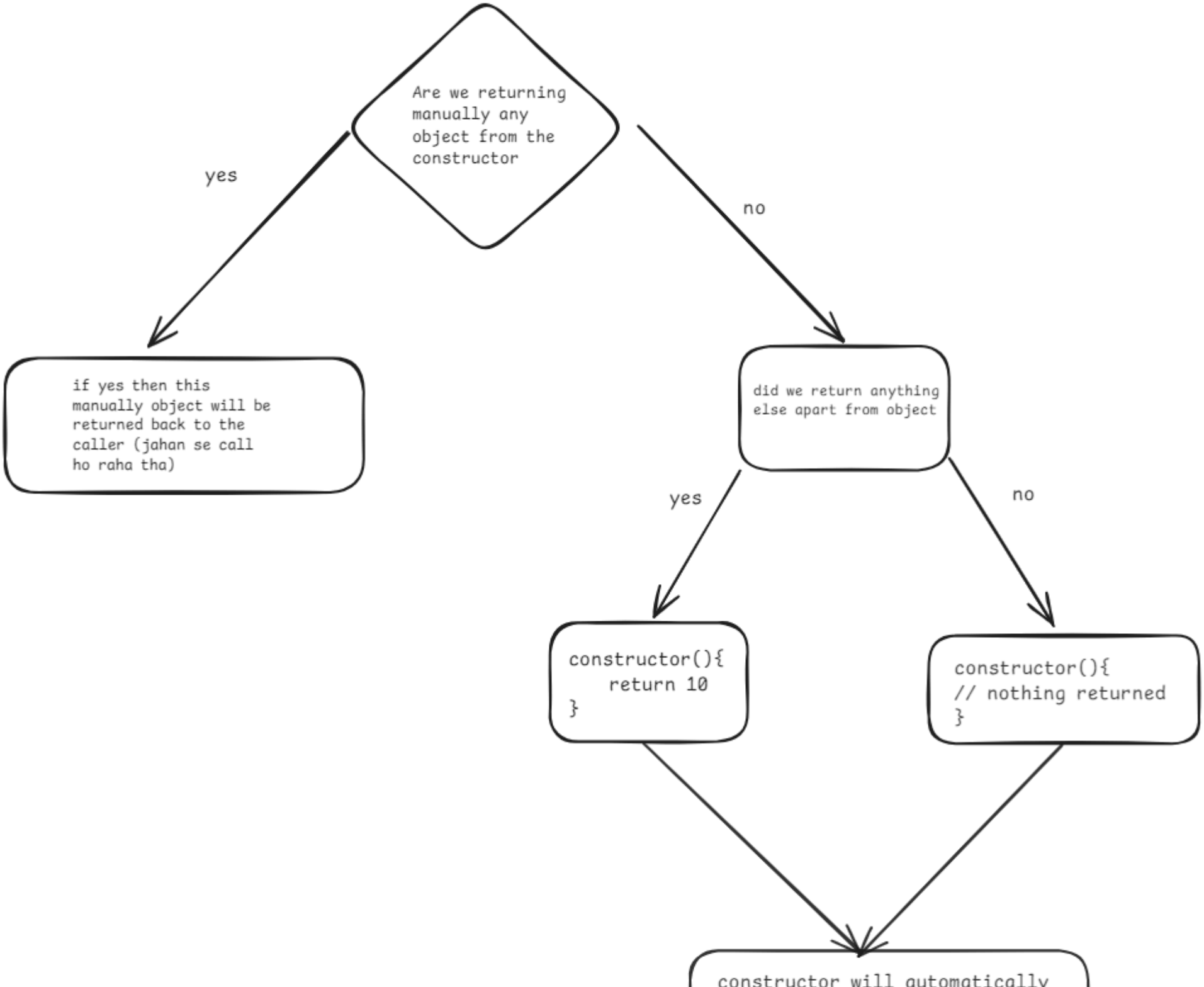
```
class Product{
        name;
        price;
        category;
        description;
        rating;

        constructor(){
                return{x:10}
        }
```

```
        addToCart(){
                console.log('Product added to the cart')


}

let vivo = new Product() // object created
console.log(vivo) // x:10
```

2nd Example what will I get if I manually return nothing from the constructor

```
class Product{
        name;
        price;
        category;
        description;
        rating;

        constructor(){
                console.log('contuctor called')
        }

        addToCart(){
                console.log('Product added to the cart')

}

let vivo = new Product() // object created
console.log(vivo)

Output:
-------
constructor called
1. Product {name: undefined, price: undefined, category: undefined, description: undefined, rating: undefined}
```

```
1. category: undefined
2. description: undefined
3. name: undefined
4. price: undefined
5. rating: undefined
6. [[Prototype]]: Object
```


Are we returning manually any object from the constructor
yes
no
if yes then this manually object will be returned back to the caller (jahan se call ho raha tha)
did we return anything else apart from object
yes
no
constructor(){
return 10
}
constructor(){
// nothing returned
}
constructor will automatically

return the value of this keyword
which is the Product

this keyword ke andar new Product
hai na toh usko hi return karega

Example:

```
class Product{
                constructor(productName, productPrice, productCategory, productDescription,productRating){

        this.name = productName
        this.price = productPrice
        this.category = productCategory
        this.description = productDescription
        this.rating = productRating

    }
   }
let vivo = new Product('Vivo V40',41000,'Mobile','Sleek Design and Flagship Phone','4.9')

console.log(vivo)

Output:
-------
1. category: "Mobile"
2. description: "Sleek Design and Flagship Phone"
3. name: "Vivo V40"
4. price: 41000
5. rating: "4.9"
```

This Keyword(this)

This bahut sare languages mein hai but har ek language mein this same kam nahi karta hai sabkla allag allag hai.
In JS this mainly depends upon call site aur aab mein bataunga ki calll site kya hai.hai This har place mein allag allag se kam karega aggar aap object mein this ko use karoge toh wahan kam karne ka tarika allag aggar aap arrow function mein pass karoge toh wahan bhi allag hai aggarvaap global mein use karna chahoge wahan bhi allag hai function ke andar aggar aap define karna chahoge wahan bhi allag tarike se kam karta hai so lets discuss each and every thing one by one.

This in Global Space

In Global space this points out to the window object

```
console.log(this)
```

In Global Space this represent to the global Object, In case of browsers the global object is window

But if we go in `nodejs` The value of global Object will be global, So if somebody ask you what is the value of this in global space then you can just say that it is global Object and that global Object may be , It can be `WINDOW` it can be `GLOBAL` where we are running the peice of code.

This in Function

```
function getData(){
        console.log(this)
}
```

Here also we will get `window` object but wait in global space also we are getting `window` in function also we are getting `window` ,

But here is a catch that both work very differently in strict mode, yes it is absolutely correct that in strict mode `this` inside the function will give us `undefined` but without strict mode it give us the window object.

so for function the value of this is actually depend upon the strict mode. and non-strict mode.

This Inside Arrow Function:

This doesn't work inside the arrow function.

```
let obj = {
        x:10,
        y:20,
        fn : () => {
                console.log(this.x, this.y)
        }
}

obj.fn()  // undefined undefined
```

The answer will be undefined so `this` doesn't work inside the arrow function, but this thing is being resolved using lexical scoping.

and what lexical scoping is , if you don't find this in the current scope then just go one level upper and so on.

let's get into an example for clear understanding.

```
let obj = {
        x:30,
        y:40,
        fun : function() {
                const fact = () => {
                        console.log(this.x, this.y)
                }
                fact()
        }
}
obj.fun()
```

Saket Sir Explanation:

1. In `this` Code is this defined in the scope of arrow function.? No
2. We Go one Scope up, i.e inside the fun function.
3. Is `this` defined inside the fun function.? Yes we have a definition of this inside the call site.

4. Who is the call site .? Obj object is the call site.
5. Hence `this` refers to obj Object and when we call arrow function we get output as 10 20

Explanation in My Own Words:
so fact is an arrow function inside the fun function, now lets see how the lexical thing is happening

when `obj.fun()` this line will execute now the control will come inside the fun function where we have defined an arrow function and inside the arrow function I am trying to access x and y value with the help of this but as we know this is not available inside the fact arrow function

Though I Have created an arrow function inside the normal function(fun()) so the lexical thing can happen due to which it will go inside the fun and there it will try to find x and y along with this . Though I haven't defined this inside the fun function but still `this` is available inside the fun function because fun is an normal function and inside the normal function this is defined

Note:
Only the this is not available inside the arrow function that's why I am unable to access x and y. window object is a global object jo browser mein chalne wali javascript mien available hota hain