🟪 The `static` keyword in JavaScript is used to define static methods and properties for a class
🟪 Static members (properties and methods) are called without instantiating their class and cannot be called through a class instance

```
class Person {
    //Properties
    static first_name='Jhon'
    static last_name='Dee'
    //method
    static getName() {
        return (`The name of the person is ${this.first_name} ${this.last_name}`)
    }
}
console.log(Person.getName());
```

Why we use `static`

🟪 Shared Properties
Static properties are used when you want to share a single piece of data across all instances of a class.

🟪 Utility Functions
Static methods are often used for utility functions that do not require the context of an instance to operate. For example, if you have a class `Circle`, you might have a static method `calculateArea` that takes a radius as a parameter and calculates the area without needing to create an instance of `Circle`

🟪 Memory Efficiency
Since static properties are not duplicated across instances, they can be more memory efficient when the same value or resource is used by many instances.

🟪 Performance
Methods that don't require access to instance data are sometimes made static so that they won't need to go through the prototype chain to be resolved. This can offer slight performance benefits.

#JavaScript_OOP