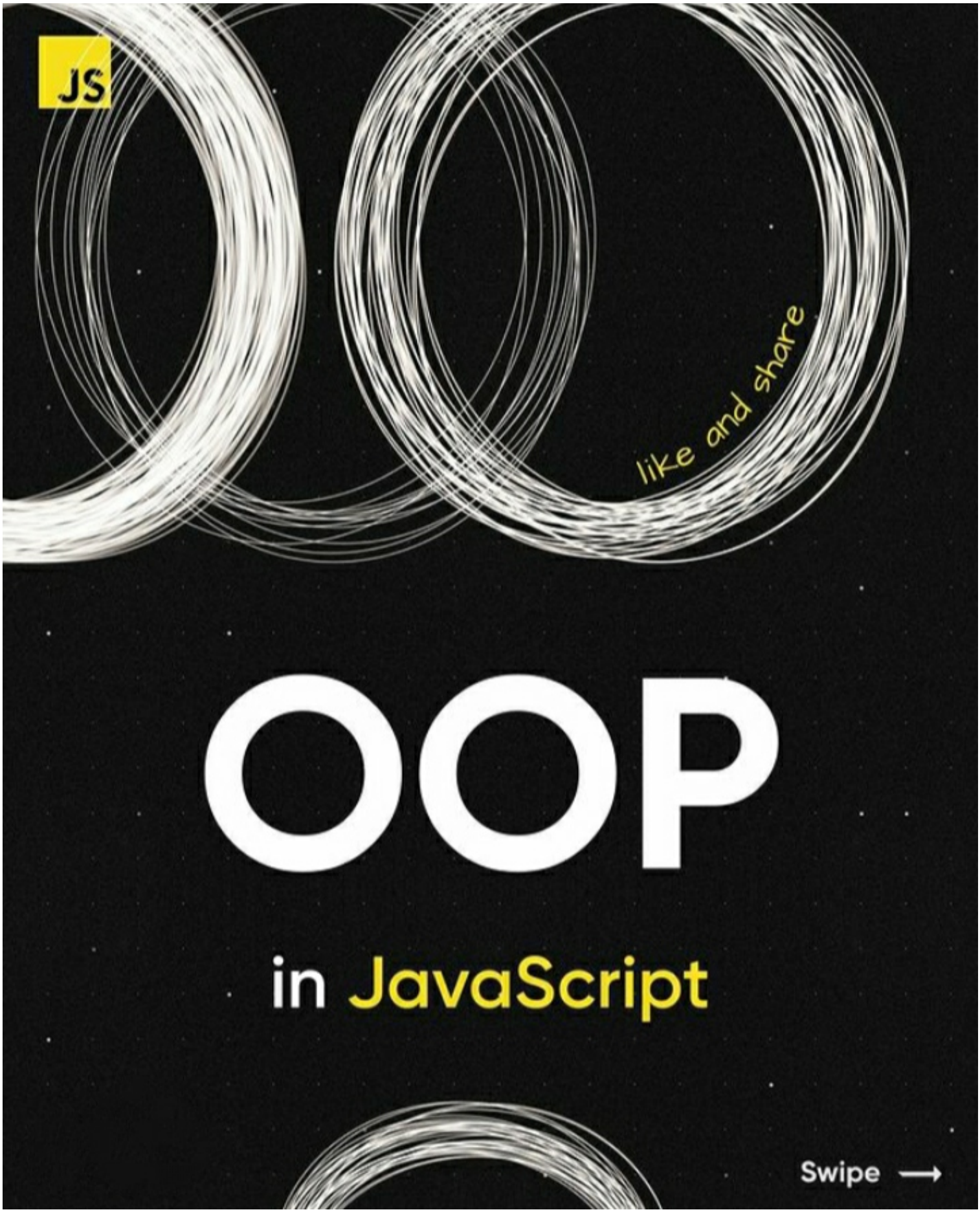
JS
like and share
OOP
in JavaScript
Swipe

JS

Class

Classes offer an improved mean of creating object templates. You can think of the class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc.

```
class Car {
  constructor(company, engine) {
    this.company = company;
    this.engine = engine;
    }

    getInfo() {
      return (
      `${this.company} has
${this.engine} engine`;
      )
    }
}
```

Here we have created a class

and added a method

Swipe ⟶

JS

First Instance

Here, we are creating **objects** of the method that we added in the class code using the **new** keyword.

```
    getInfo() {
        return (
`${this.company} has ${this.engine}
engine`
        )
    }
}

const firstCar = new Car('Bugatti', 'W16');
console.log(firstCar);
console.log(firstCar.getInfo());
const secondCar = new Car('Bently', 'V12');
console.log(secondCar);
console.log(secondCar.getInfo());
```

```
// [object Object]
{
  "company": "Bugatti",
  "engine": "W16"
}
```

```
"Bugatti has W16 engine"
```

```
// [object Object]
{
  "company": "Bently",
  "engine": "V12"
}
```

```
"Bently has V12 engine"
```

Swipe ⟶

JS

used to make the code more useful

Inheritance

Inheritance enables you to define a class that takes all the functionality from a parent class and allows you to add more.

Using class inheritance, a class can inherit all the methods and properties of another class.

```
class Sportscar extends Car {
  constructor(company, engine, doors){
    super(company, engine);
    this.doors = doors;
    }
    speed( ){
      return '267mph';
    }
}

const mySportscar = new Sportscar('Bugatti',
'W16', 2);
```

Swipe ⟶

JS

Encapsulation

a methodology used for hiding information

Implementing Encapsulation in JavaScript prevents access to the variables by adding public entities inside an object, which the callers can use to achieve specific results.

```
class Sportscar extends Car {
  #engine;
  constructor(company, engine){
    super(company, engine);
    this.#engine = engine;
  }
  getEngine(){
    return this.#engine;
  }
  setEngine(){
    this.#engine = engine;
  }
}

const mySportscar = new
Sportscar('Bugatti', 'W16', 2);
console.log(mySportscar.getEngine());
```

"W16"

Swipe ⟶

Many ↗ Change form ↗

Polymorphism

comes from the word Polymorph

Refers to the concept that there can be multiple forms of a single method, and depending upon the runtime scenario, one type of object can have different behavior. It utilizes "Inheritance" for this purpose.

```
class Sedan extends Car {
  constructor(company, engine, model){
    super(company, engine);
    this.model = model;
  }
  speed(){
  return '150mph';
  }
}

const mySportscar = new Sportscar('Bugatti',
'W16', 2);
const mySedan = new Sedan('Mercedes', 'V12',
'S-class');
console.log (mySportscar.speed()); // "267mph"
console.log (mySedan.speed()); // "150mph"
```

the speed() method is overridden in the class Sportscar and Sedan

(check page 3)

Swipe ⟶

Turn on post notification

THANKS FOR READING

FOLLOW FOR MORE AMAZING CONTENT

Like

Save