OOPs in JavaScript

What is OOP?

OOP is a way to write code using "things" called **objects**.

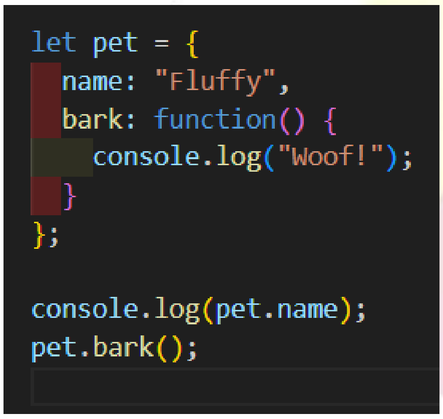
- Objects have **stuff** (like name or color).
- Objects can **do things** (like bark or move).
- Example: A toy car is an object. It has a color (red) and can roll.

Try This: Think of a pet. Write 1 thing it has (like fur) and 1 thing it does (like jump).

Objects

An **object** is like a toy. It has stuff and can-do things.

Example



```
let pet = {
  name: "Fluffy",
  bark: function() {
    console.log("Woof!");
  }
};

console.log(pet.name);
pet.bark();
```

What's Happening?

- pet is an object.
- name is stuff.
- bark is something it does.

Try This

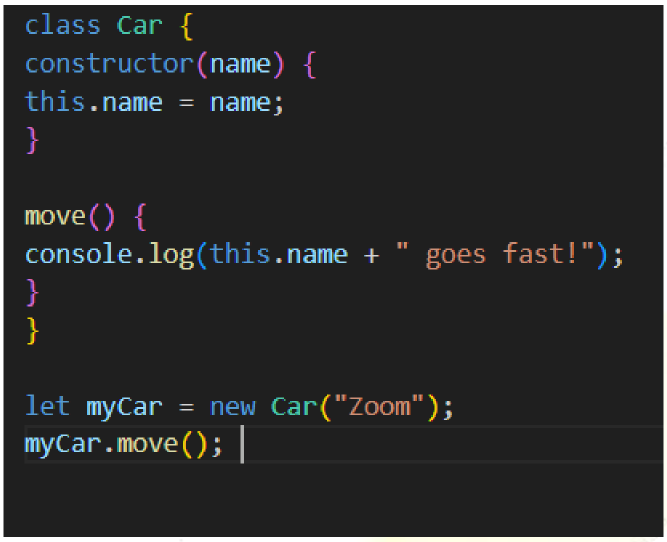
1. Open your browser and press F12 to see the console.
2. Copy the pet code and press Enter.
3. Make a toy object with:
   - Stuff: name (like "Ball").

   Do: play (shows "Let's Play!").
4. Run it and see what happens!

Classes

A **class** is like a plan to make objects. It’s like a recipe for a cake.

Example



```
class Car {
constructor(name) {
this.name = name;
}

move() {
console.log(this.name + " goes fast!");
}
}

let myCar = new Car("Zoom");
myCar.move();
```

What’s Happening?

- class Car is the plan.
- constructor adds stuff (name).
- move is something it does.
- new Car makes an object.

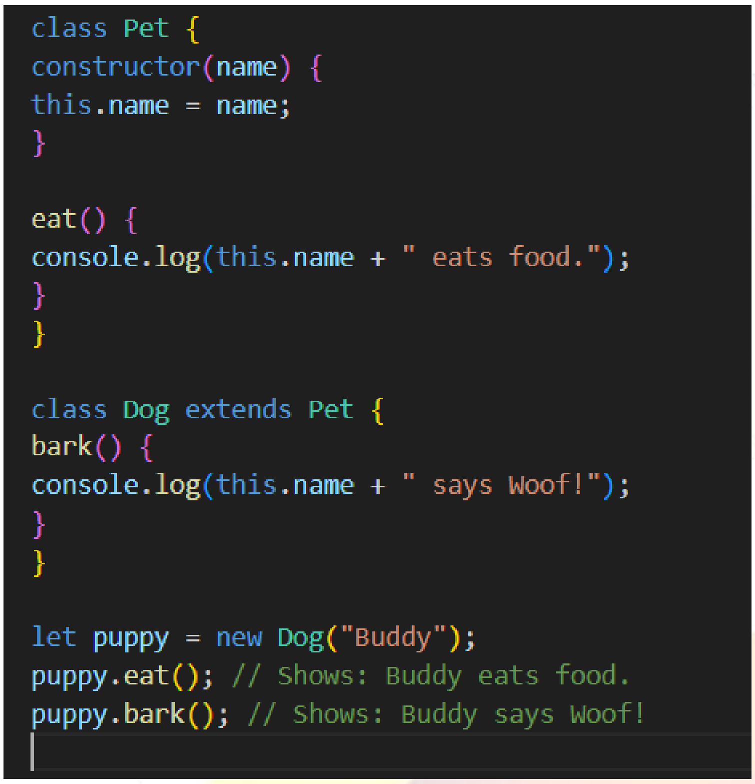
Try This

1. Copy the Car code and run it in the console.
2. Make a Kid class with:
    - Stuff: name.
    - Do: smile (shows "[name] is happy!").
3. Make a Kid object and call smile.

Inheritance

Inheritance means a class can use stuff from another class. It's like a kid getting toys from a parent.

Example



```
class Pet {
constructor(name) {
this.name = name;
}

eat() {
console.log(this.name + " eats food.");
}
}

class Dog extends Pet {
bark() {
console.log(this.name + " says Woof!");
}
}

let puppy = new Dog("Buddy");
puppy.eat(); // Shows: Buddy eats food.
puppy.bark(); // Shows: Buddy says Woof!
```

What's Happening?

- Pet is the parent.
- Dog is the kid. It gets eat and adds bark.
- extends means "use parent's stuff."

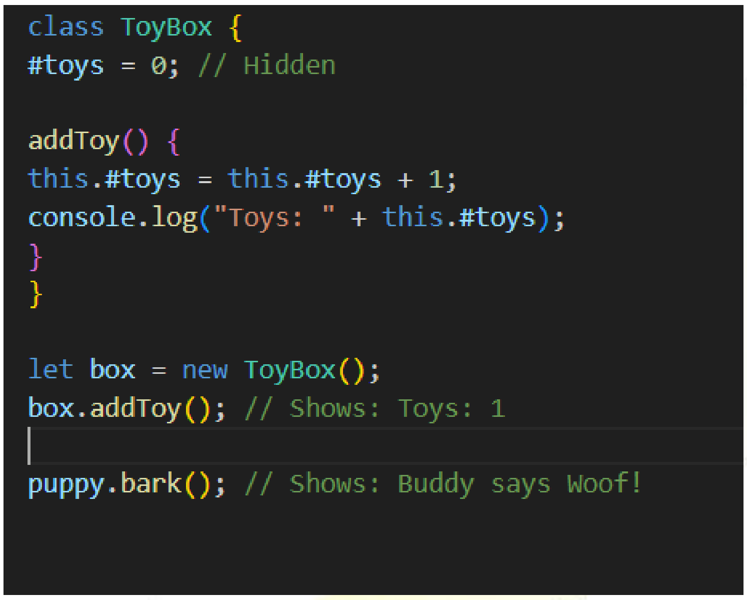
Try This

1. Copy the code and run it.
2. Make a Cat class that uses Pet.
3. Add a meow action (shows "[name] says Meow!").
4. Make a Cat object and call eat and meow.

Encapsulation

Encapsulation hides stuff. It’s like a locked toy box—you can use it but not open it.

Example



```
class ToyBox {
#toys = 0; // Hidden

addToy() {
this.#toys = this.#toys + 1;
console.log("Toys: " + this.#toys);
}
}

let box = new ToyBox();
box.addToy(); // Shows: Toys: 1

puppy.bark(); // Shows: Buddy says Woof!
```

What’s Happening?

- #toys is hidden.
- addToy adds toys safely.
- You can’t see #toys directly.

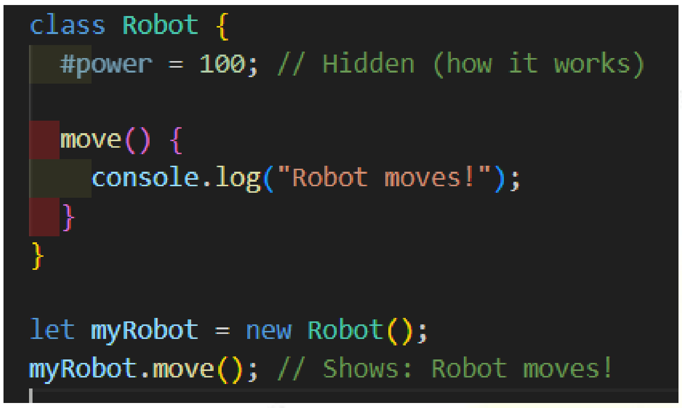
Try This

1. Copy the code and run it.
2. Make a Bag class with:
   - Hidden: #chocolates (starts at 0).
   - Do: addChocolates(add 1 candy and show total).
3. Make a Bag object and call addCandy.

Abstraction

Abstraction means showing only what's important and hiding the hard stuff. It's like using a game controller—you press buttons, but you don't see the wires inside.

Example



```
class Robot {
  #power = 100; // Hidden (how it works)

  move() {
    console.log("Robot moves!");
  }
}

let myRobot = new Robot();
myRobot.move(); // Shows: Robot moves!
```

What's Happening?

- move is easy to use.
- #power is hidden (you don't need to know how the robot works).
- Abstraction keeps things simple.

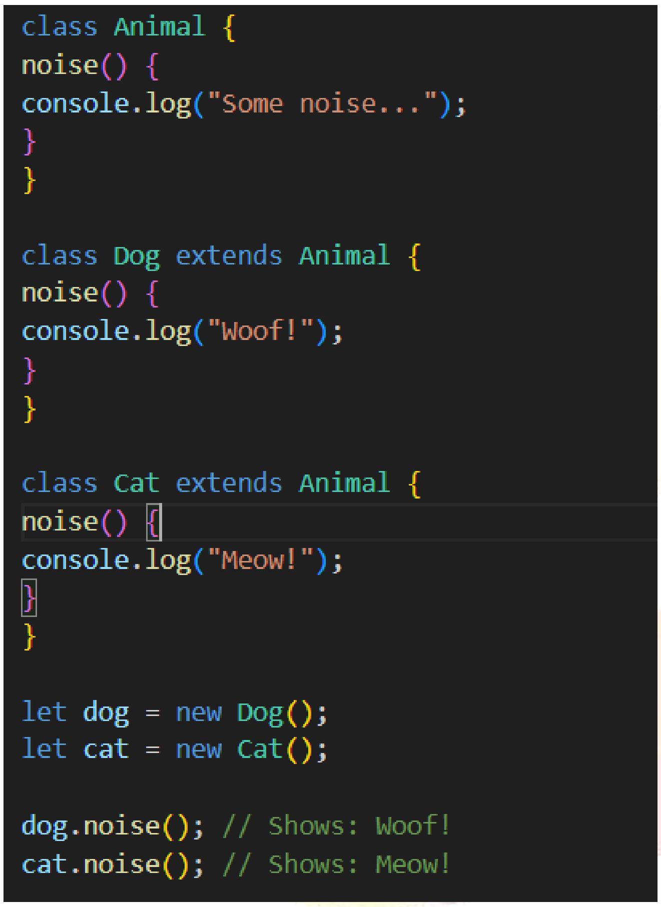
Try This

1. Copy the code and run it.
2. Make a Fan class with:
   - Hidden: #speed (starts at 0).
   - Do: spin (shows "Fan spins!").
3. Make a Fan object and call spin.

Polymorphism

Polymorphism means objects can do the same thing in different ways. It's like a "sing" button sounding different for a dog and a cat.

Example



```
class Animal {
noise() {
console.log("Some noise...");
}
}

class Dog extends Animal {
noise() {
console.log("Woof!");
}
}

class Cat extends Animal {
noise() {
console.log("Meow!");
}
}

let dog = new Dog();
let cat = new Cat();

dog.noise(); // Shows: Woof!
cat.noise(); // Shows: Meow!
```

What's Happening?

- Dog and Cat both have noise.
- Each does it differently.
- This is polymorphism!

Try This

1. Copy the code and run it.
2. Make a Bird class that uses Animal.
3. Add noise to show "Chirp!"
4. Make a Bird object and call noise.