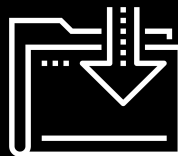




# Object-Oriented Programming (OOP)

Coding Boot Camp  
Module 10



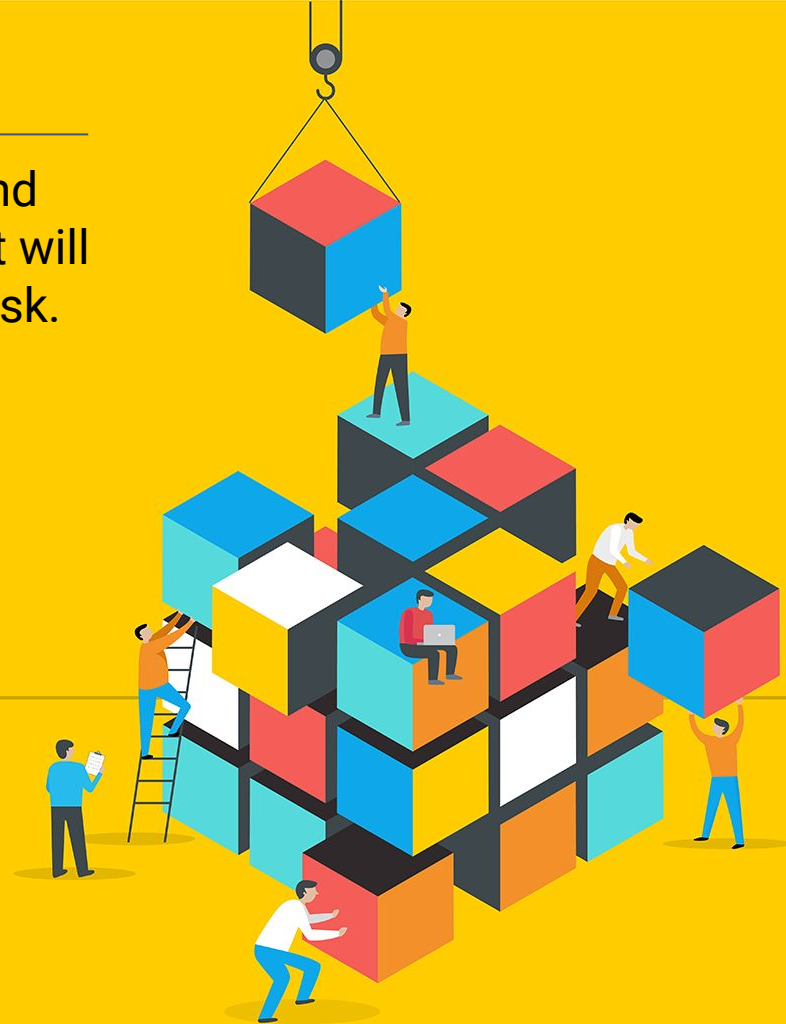


**What is programming?**

# Programming

---

**Programming** refers to designing and building an executable program that will accomplish a specific computing task. Essentially, programming is problem-solving.



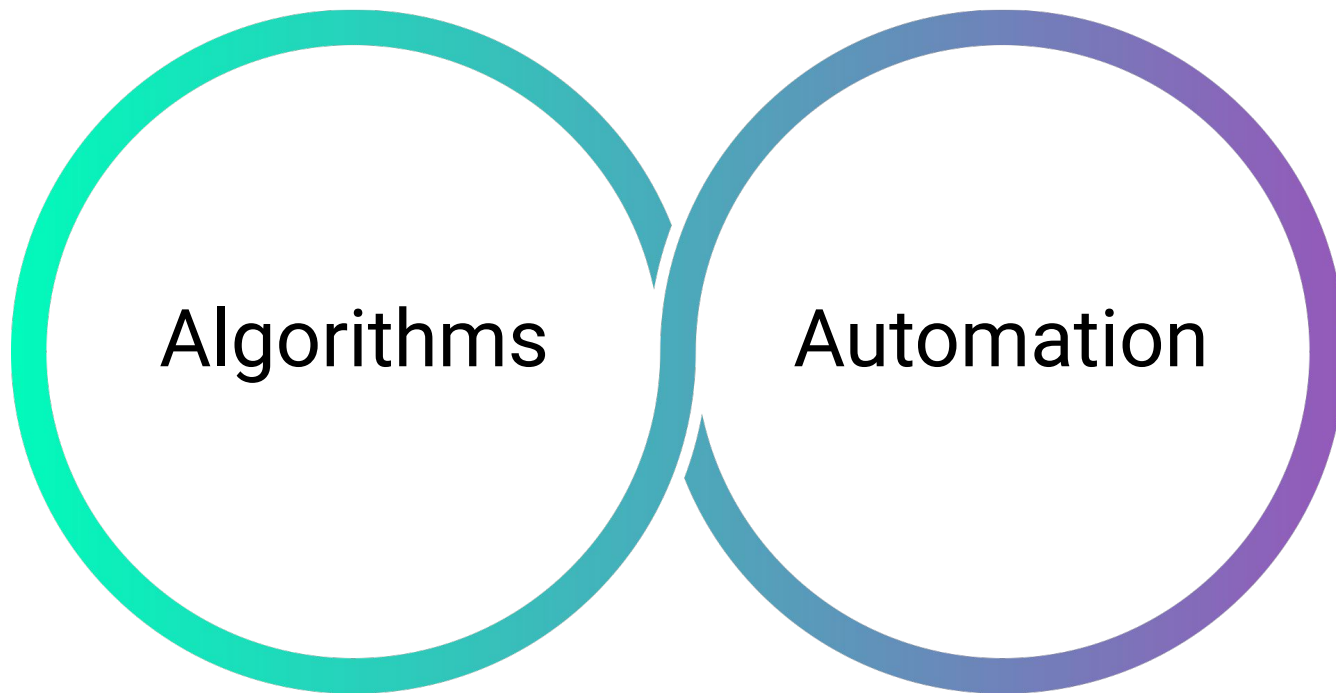


**What problems do we solve?**

# Algorithms and Automation

---

Programming enables us to solve almost any task or problem on a computer, usually in one of two primary categories: algorithms or automation.





What is DRY?

# Don't Repeat Yourself (DRY)

---

**DRY**, or **Don't Repeat Yourself**, is a fundamental programming principle. Duplicate code wastes time and memory and can confuse readers or contributors to your project.



D on't  
R epeat  
Y ourself



**What is an object?**



# Objects

---

**Objects** in JavaScript are unordered collections of related data built on a key-value structure in which values can be any data type, including functions.

```
const person = {  
  name: ['Bob', 'Smith'],  
  age: 32,  
  gender: 'male',  
  interests: ['music', 'skiing'],  
  bio() {  
    alert(  
      `${this.name[0]} ${this.name[1]} is ${this.age} years old.  
      He likes ${this.interests[0]} and ${this.interests[1]}.`  
    );  
  },  
  greeting() {  
    alert(`Hi! I'm ${this.name[0]}.`);  
  },  
};
```



**Why are objects important  
in JavaScript?**



**Because Everything in JavaScript  
Is an Object!**

---

**Well, except for primitive data types.  
Everything else is an object—essentially  
a list of key-value pairs.**

# Data types

---

## Data types that are objects

- ✓ Arrays
- ✓ Dates
- ✓ Math
- ✓ Functions
- ✓ And more!

## Primitive data types (NOT objects)

- ✗ Null
- ✗ Undefined
- ✗ Strings
- ✗ Numbers
- ✗ Symbols
- ✗ Booleans



**How do we create objects?**

# Creating Objects

---

We can use **object literals**, which define and create an object in one statement.

```
const car = { name: 'honda', model: 'civic', year: 2008, color: 'black' };
```

We can use the **new** keyword, which defines and creates a single object.

```
const Honda = new Car()
```

Or we can use **constructors**, which create objects from a blueprint.

```
class Car {  
  constructor(name, model, year, color) {  
    this.name = name;  
    this.model = model;  
    this.year = year;  
    this.color = color;  
  }  
}
```



**What is object-oriented programming?**

## **Object-Oriented Programming (OOP)**

is a programming paradigm, or pattern, centered around objects.

---

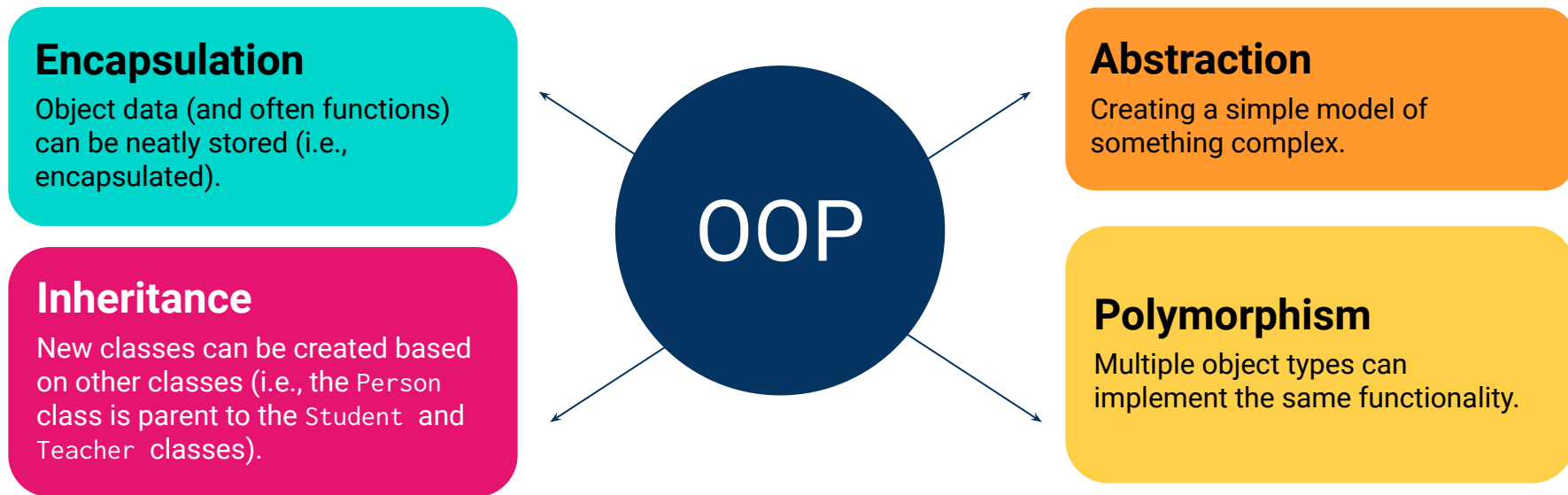
In object-oriented programming, we solve problems by employing collections of objects that work together.



# Object-Oriented Programming (OOP)

---

Their ability to communicate with each other makes objects particularly well-suited to address large, complex problems. OOP offers the following benefits:





**How can we learn to use OOP?**



**OOP is a broad concept that is best learned through real-life examples.**

---

**We begin to see the value of OOP when we use objects to model real-world things in code and provide functionality that would otherwise be hard or impossible to achieve.**

# How to Learn OOP

---

Try some of the following techniques to learn OOP:



Read the docs and practice with the provided examples.



Reverse-engineer finished code to see how it was created.



Build something from scratch.



Debug a broken app using Chrome DevTools.



And most importantly, ask questions!



# Instructor Demonstration

---

## Mini-Project