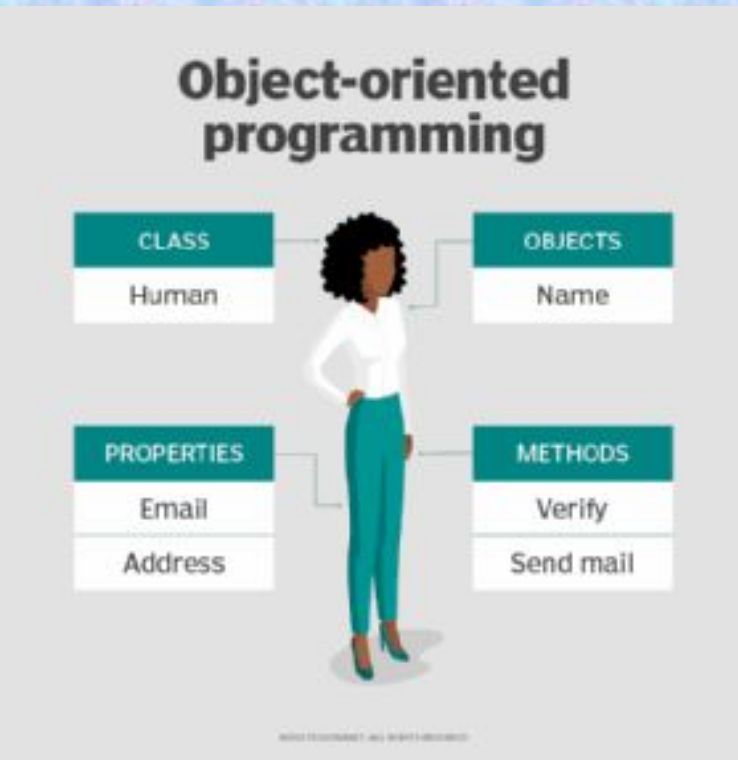


Objects

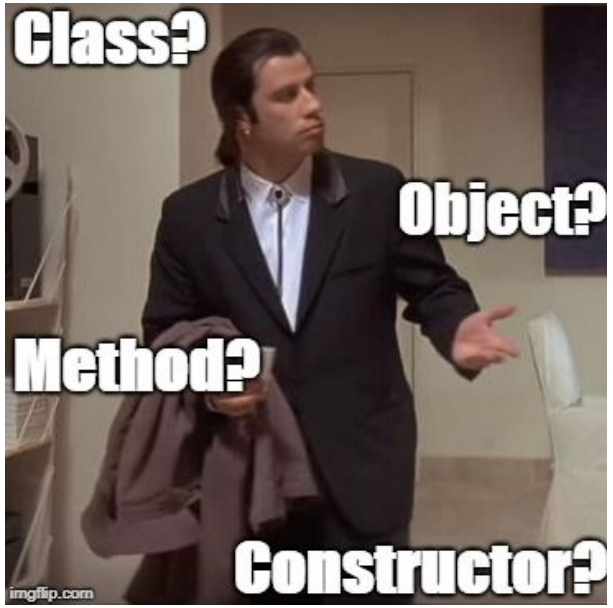
Ashutosh Jha,
Technical Officer,
C-DAC Patna

Agenda

- Introductions
- Classes and Objects
- Reference Variables and Methods
- Constructors
- Static Methods vs Instance Methods
- Reference variables as Instance of Member of the class
- String
- END

Introductions

- Object-Oriented Programming is a programming paradigm that is based on the concept of objects.
- An object is a data structure that consists of properties and methods.
- Properties represent the characteristics of an object, while methods represent the behavior of an object.
- OOP emphasizes the concept of modularity, encapsulation, and reusability.

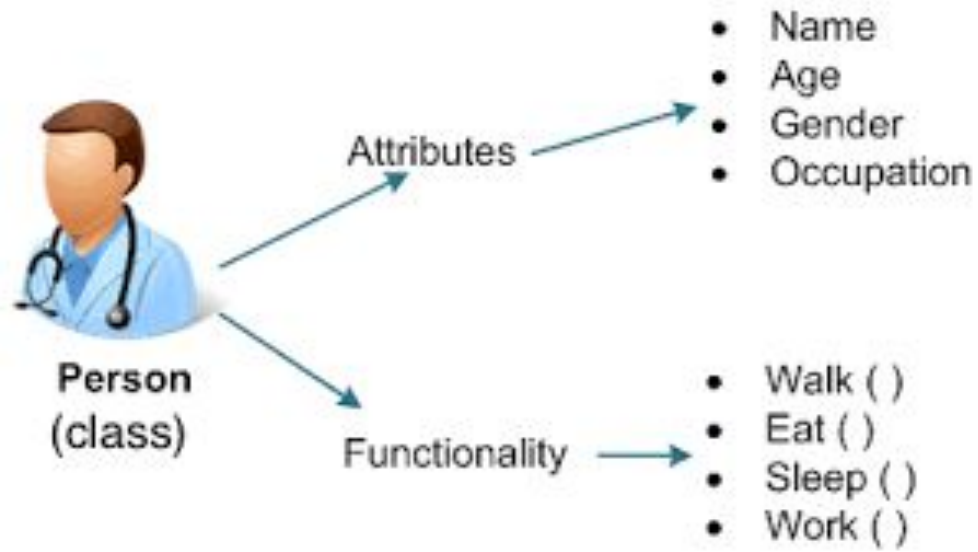


Four Pillars of Object-Oriented Programming

- The four pillars of OOP are:
  - Encapsulation: the idea of hiding data and behavior within an object.
  - Abstraction: the idea of exposing only relevant information to the user.
  - Inheritance: the idea of creating new objects based on existing objects.
  - Polymorphism: the idea of using a single interface to represent multiple objects.

Classes and Objects

- A class is a blueprint for creating objects that share the same properties and methods.
- An object is an instance of a class.
- In OOP, we use classes to create objects, which makes it easier to organize our code and reuse it.



Benefits of Object-Oriented Programming

- Object-Oriented Programming offers several benefits:
  - Modularity: the ability to break down a program into smaller, more manageable parts.
  - Reusability: the ability to reuse code in different parts of a program or in different programs.
  - Maintainability: the ability to make changes to a program without affecting other parts of the program.
  - Scalability: the ability to add new features or functionality to a program without affecting the existing code.

Reference Variables

- A reference variable is a variable that holds the memory location of an object.
- In JavaScript, objects are created using constructor functions or object literals.
- To create a reference variable, we use the var, let, or const keyword followed by the variable name, an equals sign, and the object constructor or object literal.

Example

- Here's an example of creating a reference variable using an object literal:

```
let person = {
  firstName: "John",
  lastName: "Doe",
  age: 30
};
```

- Here's an example of creating a reference variable using an object constructor:

```
function Person(firstName, lastName, age) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.age = age;
}

let person = new Person("John", "Doe", 30);
```

Reference Methods

- A method is a function that belongs to an object and can be called on that object.
- Methods are used to perform actions on objects or to return information about objects.
- In JavaScript, methods are defined inside the object using the this keyword.

Example

- Here's an example of defining a method in an object using the this keyword:

```
let person = {
  firstName: "John",
  lastName: "Doe",
  age: 30,
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
};
```

- Here's an example of calling the method on the object:

```
console.log(person.fullName()); // Output: "John Doe"
```

Constructors

- Constructors are special methods that are used to initialize objects of a class. They have the same name as the class and are called automatically when an object is created. There are two types of constructors: default constructors and parameterized constructors.
- Default Constructor
  - A default constructor is a constructor that has no parameters. It is called automatically when an object is created.
  - In JavaScript, you can create a default constructor using the "constructor" keyword. Here is an example:

Default Constructor

- A default constructor is a constructor that has no parameters. It is called automatically when an object is created.
- In JavaScript, you can create a default constructor using the "constructor" keyword. Here is an example:

```
class Person {
    constructor() {
        this.name = "John";
        this.age = 30;
    }
}

let person = new Person();
console.log(person.name); // Output: John
console.log(person.age); // Output: 30
```

Parameterized Constructor

- A parameterized constructor is a constructor that has one or more parameters. It is used to set initial values for object properties when an object is created.
- In JavaScript, you can create a parameterized constructor using the "constructor" keyword and passing in parameters. Here is an example:

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
}

let person = new Person("John", 30);
console.log(person.name); // Output: John
console.log(person.age); // Output: 30
```

Static Method

- A static method is a method that belongs to the class and not to the object. It can be called using the class name without creating an object.
- In JavaScript, you can create a static method using the "static" keyword. Here is an example:

```
class Calculator {
  static add(num1, num2) {
    return num1 + num2;
  }
}

console.log(Calculator.add(5, 10)); // Output: 15
```

Instance Method

- An instance method is a method that belongs to the object and not to the class. It can be called using the object name after creating an object.
- In JavaScript, you can create an instance method by defining a method inside the class. Here is an example:

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  sayHello() {
    console.log("Hello, my name is " + this.name);
  }
}
let person = new Person("John", 30);
person.sayHello(); // Output: Hello, my name is John
```

Reference Variable as Instance Member of the Class

- Reference variables can also be used as instance members of a class in JavaScript.
- When a reference variable is used as an instance member, it is assigned to an object of a different class.
- The reference variable can then be used to access the properties and methods of the object.
- In the above example, "car" is a reference variable that is used as an instance member of the "Person" class. It is assigned to an object of the "Car" class, and the "getCarBrand" method is used to access its "brand" property.

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

class Person {
  constructor(name, car) {
    this.name = name;
    this.car = car;
  }

  getCarBrand() {
    return this.car.brand;
  }
}

let car = new Car("Toyota");
let person = new Person("John", car);
console.log(person.getCarBrand()); // Output: Toyota
```

String

The String class is used to create and manipulate strings in JavaScript.

Strings are a sequence of characters that can include letters, numbers, and symbols.

The String class provides a number of properties and methods that can be used to manipulate strings.

```
let str = "Hello, world!";
console.log(str.length); // Output: 13
console.log(str.toUpperCase()); // Output: HELLO, WORLD!
console.log(str.indexOf("world")); // Output: 7
```

Write your own program

- Write a program that creates a "Rectangle" class with width and height properties, and a method that calculates and returns the area of the rectangle. Then create a new object of the "Rectangle" class and use it to print the area.

Write your own program

- Write a program that creates a "Student" class with a name and a list of subjects properties. The subjects property should be an array of objects with name and grade properties. The class should also have a method that calculates and returns the average grade of all subjects. Then create a new object of the "Student" class and use it to print the average grade.

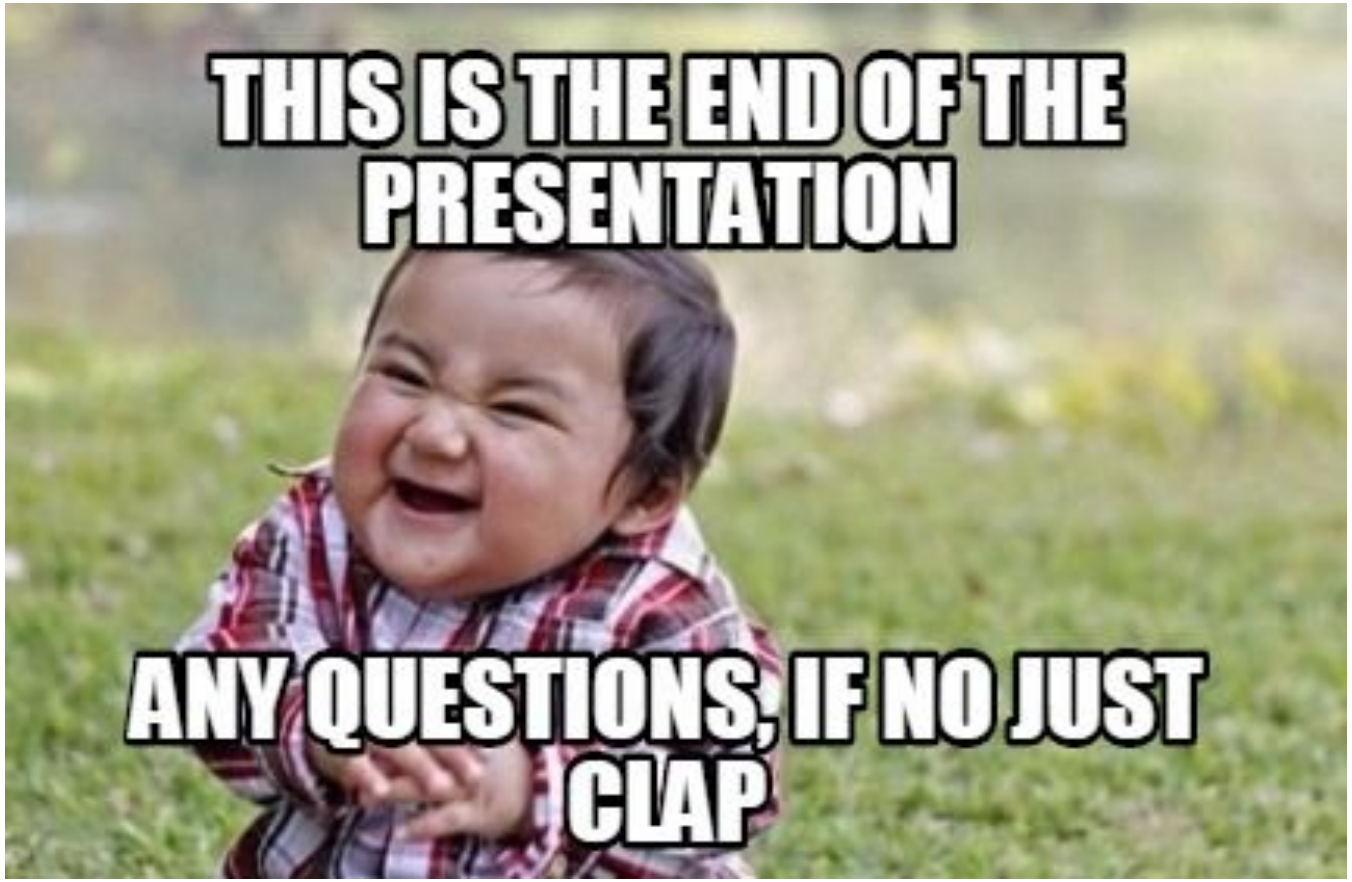
Recap

- Reference variables and methods are important for accessing and manipulating objects in JavaScript.
- Constructors provide a way to initialize objects of a class.
- Static methods belong to the class, while instance methods belong to the object.
- Reference variables can be used as instance members to access properties and methods of another class.
- The String class provides a number of properties and methods for manipulating strings in JavaScript.

End of Session, Now What to do?

- Practice implementing these concepts in your own programs.
- Continue to explore object-oriented programming.
- Stay curious and keep learning!

Questions



Thank You!!!

https://www.linkedin.com/company/c-dac-patna

https://twitter.com/CDAC_Patna

https://www.facebook.com/PatnaCDAC/

https://www.youtube.com/c/CDACPatna