JavaScript 面向对象

类和对象

属性和方法

继承

多态

封装

类和对象

> 类是用于定义对象的模板或蓝图；它包含对象的属性和方法，我们可以使用class关键字来定义类。

JavaScript

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

  sayHello() {
    console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);
  }
}
```

> 在上面的例子中，我们定义了一个名为Person的类，它有两个属性：name和age，以及一个方法sayHello，构造函数constructor用于初始化对象的属性。

> 对象是类的一个实例，我们可以使用new关键字来创建对象。

JavaScript

```
let person1 = new Person('Alice', 25);
```

> 在上面的例子中，我们创建了一个名为person1的Person对象，并将其属性初始化为Alice和25。

属性和方法

> 属性是对象的特征或状态。它们用于描述对象的特点。我们可以在构造函数中初始化对象的属性。

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

> 在上面的例子中，我们定义了一个Person类，并在构造函数中初始化了两个属性：name和age。

> 方法是对象的行为或操作。它们用于描述对象的行为。在类中，我们可以定义一组方法，以便对对象执行不同的操作。

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

  sayHello() {
    console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);
  }
}

```

> 在上面的例子中，我们定义了一个sayHello方法，它用于输出对象的属性。

继承

> 继承是面向对象编程的一个重要概念。它允许我们从现有的类创建新类，从而在不重复代码的情况下扩展现有的功能。在JavaScript中，我们可以使用extends关键字来创建一个新类，它从现有的类继承属性和方法。

```javascript
class Student extends Person {
  constructor(name, age, grade) {
    super(name, age);
    this.grade = grade;
  }

  sayHello() {
    console.log(`Hello, my name is ${this.name}, I am ${this.age} years old, and I am in grade ${this.grade}.`);
  }
}
```

> 在上面的例子中，我们定义了一个名为Student的新类，它从现有的Person类继承了name和age属性和sayHello方法。Student类还有一个名为grade的新属性，并覆盖了sayHello方法以添加新信息。

多态

> 多态是面向对象编程的另一个重要概念。它允许不同的类实现相同的方法，以便在不同的情况下以不同的方式处理相同的请求。在JavaScript中，我们可以使用相同的方法名在不同的类中实现不同的行为。

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

  speak() {
    console.log(`${this.name} makes a noise.`);
  }
}

class Dog extends Animal {
  speak() {
    console.log(`${this.name} barks.`);
  }
}

class Cat extends Animal {
  speak() {
    console.log(`${this.name} meows.`);
  }
}

const animals = [
  new Dog('Fido'),
  new Cat('Fluffy'),
  new Dog('Max'),
  new Cat('Whiskers')
];

animals.forEach(animal => {
  animal.speak();
});

```

> 在上面的例子中，我们定义了一个名为Animal的基类，并从中派生出Dog和Cat类。这两个类都实现了Animal的speak方法，并在方法中以不同的方式输出信息。最后，我们创建了一些Dog和Cat的实例，并在它们上面调用speak方法，以验证它们会输出不同的信息。

封装

> 当使用面向对象编程时，我们通常会封装数据和方法以确保对象的安全性和可维护性。在JavaScript中，封装是通过创建类和使用访问修饰符来实现的。

JavaScript

```javascript
// 创建一个类
class Person {
  // 声明私有属性
  #name;
  #age;

  // 构造函数，用于初始化对象
  constructor(name, age) {
    this.#name = name;
    this.#age = age;
  }

  // 声明公有方法，用于访问私有属性
  getName() {
    return this.#name;
  }

  getAge() {
    return this.#age;
  }
}

// 创建一个Person对象
const person = new Person('张三', 18);

// 访问私有属性（会报错）
console.log(person.#name); // Uncaught SyntaxError: Private field '#name' must be declared in an enclosing class

// 访问公有方法
console.log(person.getName()); // '张三'
console.log(person.getAge()); // 18

```