

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。

```
1 class Person {  
2   constructor(name, age) {  
3     this.name = name;  
4     this.age = age;  
5   }  
6  
7   sayHello() {  
8     console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);  
9   }  
10 }
```

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。

```
1 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。

```
1 class Person {  
2   constructor(name, age) {  
3     this.name = name;  
4     this.age = age;  
5   }  
6 }
```

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。

```
1 class Person {  
2   constructor(name, age) {  
3     this.name = name;  
4     this.age = age;  
5   }  
6  
7   sayHello() {  
8     console.log(`Hello, my name is ${this.name} and I am ${this.age} year  
9     s old.`);  
10  }  
11 }
```

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。

```
1 class Student extends Person {  
2   constructor(name, age, grade) {  
3     super(name, age);  
4     this.grade = grade;  
5   }  
6  
7   sayHello() {  
8     console.log(`Hello, my name is ${this.name}, I am ${this.age} years old,  
9     and I am in grade ${this.grade}.`);  
10  }  
}
```

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。

```
1 class Animal {
2   constructor(name) {
3     this.name = name;
4   }
5
6   speak() {
7     console.log(`${this.name} makes a noise.`);
8   }
9 }
10
11 class Dog extends Animal {
12   speak() {
13     console.log(`${this.name} barks.`);
14   }
15 }
16
17 class Cat extends Animal {
18   speak() {
19     console.log(`${this.name} meows.`);
20   }
21 }
22
23 const animals = [
24   new Dog('Fido'),
25   new Cat('Fluffy'),
26   new Dog('Max'),
27   new Cat('Whiskers')
28 ];
29
30 animals.forEach(animal => {
31   animal.speak();
32 });
33
```

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。

```
1 // 创建一个类
2 class Person {
3     // 声明私有属性
4     #name;
5     #age;
6
7     // 构造函数，用于初始化对象
8     constructor(name, age) {
9         this.#name = name;
10        this.#age = age;
11    }
12
13    // 声明公有方法，用于访问私有属性
14    getName() {
15        return this.#name;
16    }
17
18    getAge() {
19        return this.#age;
20    }
21 }
22
23 // 创建一个Person对象
24 const person = new Person('张三', 18);
25
26 // 访问私有属性（会报错）
27 console.log(person.#name); // Uncaught SyntaxError: Private field '#name'
    must be declared in an enclosing class
28
29 // 访问公有方法
30 console.log(person.getName()); // '张三'
31 console.log(person.getAge()); // 18
32
```