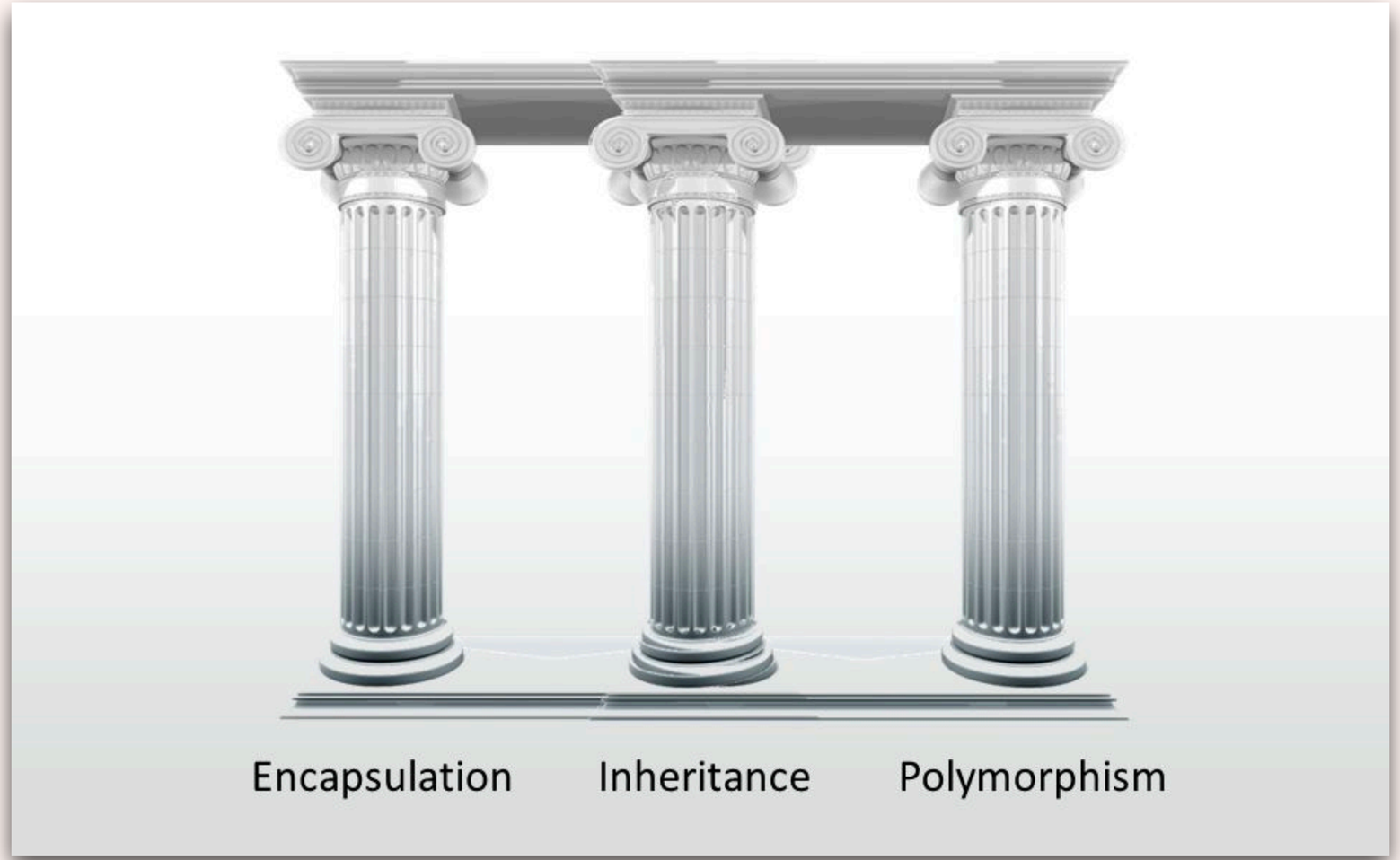
JAVASCRIPT

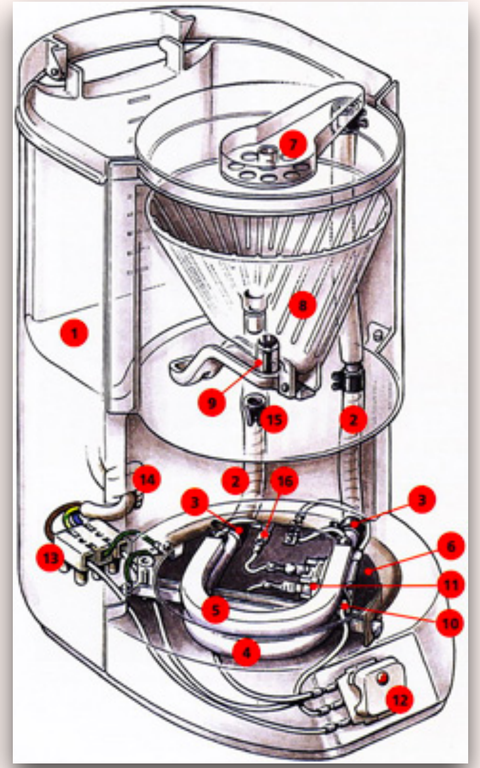
THREE PILLARS



INTERNAL AND EXTERNAL INTERFACES

In object-oriented programming, properties and methods are divided into 2 groups:

- **Internal interface** - methods and properties accessible from other methods of the class, but not outside the class.
- **External interface** - methods and properties available from outside the class.



PRIVATE PROPERTIES

- All properties of class are public by default.
- When you declare any property of a class as private, you can only access it in that class.
- The syntax for creating private fields is simple, just use the **#** before the property name.
- Remember to use **#** when you need to access such a property.

```
class Foo {
  #privateValue = 42;
  static getPrivateValue(foo) {
    return foo.#privateValue;
  }
}


Foo.getPrivateValue(new Foo()); // >> 42
```

ENCAPSULATION

- If you are using an instance of a class, you cannot reference the private fields of that class. You can only refer to private fields within the class that defines them.
- It can also mean hiding the internal implementation from other components.

```
class Foo {
  #bar;
  method() {
    this.#bar; // Работает
  }
}
let foo = new Foo();
foo.#bar; // Неверно!
```

POLYMORPHISM

- Ability of an object during its execution to refer to instances of its own class or any inherit class.
- Inheriting classes can override the method.

```
class Person {
  constructor(name) {
    this.name = name;
  }
  me() {
    return `My name is ${this.name}`;
  }
}
const axel = new Person('Axel');
console.log(axel.me());
  // ->  'My name is Axel'
class Employee extends Person {
  constructor (name, salary) {
    super(name);
    this.salary = salary;
  }
  me() {
    return `My name is ${this.name} and my salary is
${this.salary}`;
  }
}


const nick = new Employee('Nick', 3000);
console.log(nick.me());
  // ->  'My name is Nick and my salary is 3000'
```

PROTECTED PROPERTIES

- Protected properties usually start with the _ prefix.
- Can be accessed by inheritors of the parent class.
- It's just an agreement - everything is still visible from the outside, but we just agreed with ourselves about the following: everything that starts with an underline is not used from the outside. We can, but we won't.

```
class CoffeeMachine {
  _waterAmount = 0;

  set waterAmount(value) {
    if (value < 0) throw new Error("Отрицательное количество воды");
    this._waterAmount = value;
  }

  get waterAmount() {
    return this._waterAmount;
  }

  constructor(power) {
    this._power = power;
  }

}

// создаём новую кофеварку
let coffeeMachine = new CoffeeMachine(100);

// устанавливаем количество воды
coffeeMachine.waterAmount = -10; // Error: Отрицательное количество воды
```

READ-ONLY PROPERTIES

```
class CoffeeMachine {
  // ...

  constructor(power) {
    this._power = power;
  }

  get power() {
    return this._power;
  }

}

// создаём кофеварку
let coffeeMachine = new CoffeeMachine(100);

alert(`Мощность: ${coffeeMachine.power}W`); // Мощность: 100W

coffeeMachine.power = 25; // Error (no setter)
```