30. 构造函数与 new 的机制

1. 核心概念 (Core Concept)

在 JavaScript 中，构造函数（Constructor Function）是一种特殊的函数，通常用于创建和初始化对象实例。`new` 运算符是用于调用构造函数，从而创建一个新的对象实例并执行构造函数内部的代码来对其进行初始化。

2. 为什么需要它？ (The "Why")

- **实例化对象：** 允许我们基于一个通用的“模板”（构造函数）创建多个具有相同结构和初始状态的对象实例。
- **代码复用：** 将对象的初始化逻辑封装在构造函数中，避免重复编写创建相似对象的代码。
- **面向对象编程基础：** 是实现基于原型的继承以及模拟类的重要机制。

3. API 与用法 (API & Usage)

构造函数本身是标准函数，但其特殊之处在于它们旨在配合 `new` 运算符使用。

构造函数的定义：

通常使用帕斯卡命名法（Pascal Case，首字母大写）来区分构造函数与普通函数，但这并非强制语法要求，而是一种约定俗成的最佳实践。

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

// 可以为构造函数的 prototype 添加方法，所有实例共享这些方法
Person.prototype.sayHello = function() {
  console.log(`Hello, my name is ${this.name} and I am ${this.age} years
old.`);
};
```

`new` 运算符的使用：

当使用 `new` 运算符调用构造函数时，会发生以下几个步骤：

1. **创建一个新的空对象：** 一个全新的、空的、普通 JavaScript 对象被创建。
2. **设置新对象的原型链：** 新创建的对象的 `[[Prototype]]` 内部属性（可以通过 `__proto__` 或 `Object.getPrototypeOf()` 访问）会被链接到构造函数的 `prototype` 对象。这意味着新对象可以访问构造函数 `prototype` 上定义的方法和属性。

3. **将 `this` 绑定到新对象：** 在构造函数内部，`this` 关键字会被绑定到这个新创建的对象。这意味着在构造函数内部使用 `this` 可以为这个新对象添加属性和方法。
4. **执行构造函数内的代码：** 构造函数体内的代码会被执行，对新对象进行初始化（例如设置属性值）。
5. **返回值：**
   - 如果构造函数没有显式地 `return` 语句，或者 `return` 的是一个非对象基本类型值（如字符串、数字、布尔值、null、undefined），则 `new` 表达式会返回步骤 1 中创建的新对象。
   - 如果构造函数**显式地 `return` 了一个对象**（包括数组、函数、通过 `{}` 或 `new Object()` 创建的对象等），则 `new` 表达式会返回这个显式返回的对象，而不是步骤 1 中创建的新对象。

代码示例：

```
// 使用 new 运算符调用 Person 构造函数创建实例
const person1 = new Person("Alice", 30);
const person2 = new Person("Bob", 25);

console.log(person1); // 输出: Person { name: "Alice", age: 30 } (或类似结构)
console.log(person2); // 输出: Person { name: "Bob", age: 25 } (或类似结构)

person1.sayHello(); // 输出: Hello, my name is Alice and I am 30 years old.
person2.sayHello(); // 输出: Hello, my name is Bob and I am 25 years old.

// 示例：构造函数显式返回一个对象
function Creator() {
  this.value = 1;
  return { value: 2 }; // 显式返回一个对象
}

const instanceWithReturn = new Creator();
console.log(instanceWithReturn.value); // 输出: 2 (返回了显式指定的对象)

// 示例：构造函数显式返回一个非对象值
function Creator2() {
  this.value = 1;
  return 100; // 显式返回一个基本类型值
}

const instanceWithReturnBasic = new Creator2();
console.log(instanceWithReturnBasic.value); // 输出: 1 (返回了新创建的对象)
console.log(instanceWithReturnBasic); // 输出: Creator2 { value: 1 } (或类似结构)
```

4. 关键注意事项 (Key Considerations)

- **忘记使用 `new`：** 如果调用构造函数时忘记使用 `new`，它就变成一个普通的函数调用。此时，函数内部的 `this` 通常会指向全局对象（在严格模式下是 `undefined`），而不是新创建的对象。这会导致属性被添加到全局对象或引发错误，而不是添加到实例对象。
- **`this` 的绑定：** `new` 运算符是改变函数内部 `this` 指向的关键机制之一。理解 `new` 如何绑定 `this` 对于理解代码行为至关重要。
- **`prototype` 的作用：** `new` 过程中的原型链连接是实现方法的共享和继承的基础。实例通过原型链访问构造函数 `prototype` 上的属性和方法。
- **显式返回对象：** 构造函数显式返回一个对象会覆盖 `new` 默认返回新对象的机制。这是相对不常见的模式，但在某些特定场景下可能会用到，理解其行为很重要。

5. 参考资料 (References)

- **MDN Web Docs: new 运算符**
  - https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/new
- **MDN Web Docs: 构造函数**
  - 虽然没有专门的“构造函数”页面，但 MDN 关于 函数、`this` 和 原型的文档详细解释了相关概念。例如：https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Functions 和 https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/this
- **JavaScript.info: Constructor, "new" operator**
  - https://javascript.info/constructor-new (一个结构清晰、易于理解的技术博客，广泛被认可)