Каррирование (Currying) в JavaScript

Основные концепции

Каррирование — это техника преобразования функции с несколькими аргументами в последовательность функций с одним аргументом.

Ключевые особенности:

* Фиксация части аргументов
* Создание специализированных функций
* Постепенное применение аргументов

Реализация через bind()

Базовый пример умножения

javascript

Copy

Download

function multiply(a, b) {

return a \* b;

}

// Фиксация первого аргумента

const double = multiply.bind(null, 2);

console.log(double(5)); // 10

const triple = multiply.bind(null, 3);

console.log(triple(5)); // 15

Практические примеры

1. Функция приветствия

javascript

Copy

Download

function greet(gender, age, name) {

return `${gender === 'male' ? 'Уважаемый' : 'Уважаемая'} ${name}, ${age} лет`;

}

// Фиксация аргументов

const greetMale = greet.bind(null, 'male');

console.log(greetMale(30, 'Иван')); // "Уважаемый Иван, 30 лет"

const greetYoungFemale = greet.bind(null, 'female', 16);

console.log(greetYoungFemale('Анна')); // "Уважаемая Анна, 16 лет"

2. Логирование с префиксом

javascript

Copy

Download

function log(level, message) {

console.log(`[${level}] ${message}`);

}

const logError = log.bind(null, 'ERROR');

logError('Соединение прервано'); // "[ERROR] Соединение прервано"

Современные реализации (ES6+)

1. Явное каррирование

javascript

Copy

Download

const curryMultiply = a => b => a \* b;

const double = curryMultiply(2);

console.log(double(5)); // 10

2. Универсальная функция каррирования

javascript

Copy

Download

function curry(fn) {

return function curried(...args) {

return args.length >= fn.length

? fn.apply(this, args)

: (...moreArgs) => curried.apply(this, args.concat(moreArgs));

};

}

const curriedSum = curry((a, b, c) => a + b + c);

console.log(curriedSum(1)(2)(3)); // 6

Преимущества каррирования

1. **Специализация функций** - создание более конкретных версий
2. **Переиспользование кода** - избегание дублирования
3. **Композиция функций** - возможность комбинировать простые функции
4. **Отложенное выполнение** - применение аргументов по мере необходимости

Сравнение подходов

| **Подход** | **Фиксация аргументов** | **Гибкость** | **Читаемость** |
| --- | --- | --- | --- |
| bind | Да | Низкая | Средняя |
| Явное каррирование | Да | Высокая | Высокая |
| Универсальное каррирование | Да | Максимальная | Средняя |

Каррирование — мощный инструмент функционального программирования, позволяющий создавать гибкие и переиспользуемые функции.