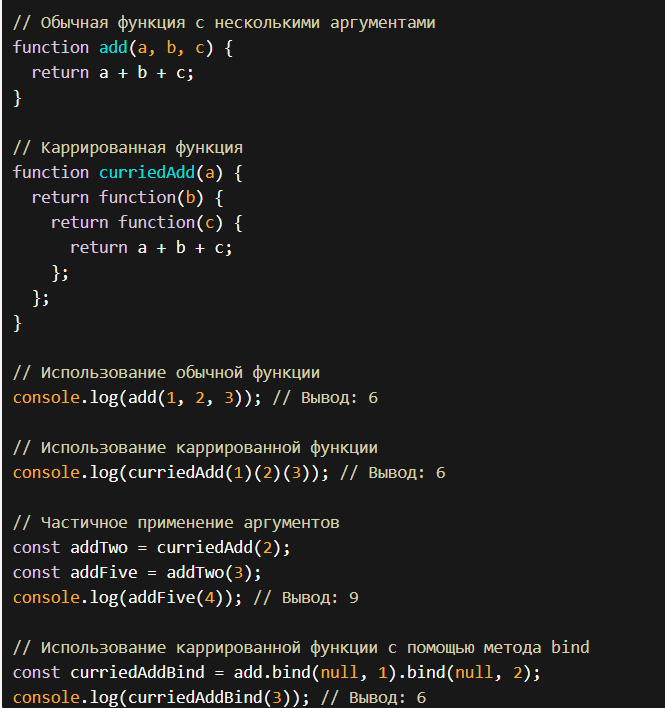
1. Каррирование (currying) – это процесс преобразования нескольких функций с несколькими аргументами в последовательность функций с одним аргументом.

Каждая функция из этой последовательности принимает один аргумент и возвращает новую функцию, которая ожидает следующий аргумент.

Этот процесс будет продолжаться до тех пор, пока все аргументы не будут переданы.

Каррированная функция – это функция, которая была преобразована с использованием каррирования. Вместо того, чтобы принимать все аргументы сразу, она принимает один аргумент за раз и возвращает новую функцию до тех пор, пока не будут переданы все аргументы. После чего она выполняет свою основную логику и возвращает результат.

Пример каррированной функции:



1. Генераторы:

Генераторы в JavaScript – это специальный тип функций, введенный в ECMAScript 2015 (ES6), который позволяет создавать итераторы\* для управления последовательными значениями.

Генераторы представляют собой мощный инструмент для работы с асинхронным кодом, обработки коллекций и ленивых вычислений.

\*итераторы--Итерируемые или, иными словами, «перебираемые» объекты – это те, содержимое которых можно перебрать в цикле. Например таким объектом является массив.

Основные характеристики генераторов:

1) **Функция-генератор:** Генератор определяется с использованием ключевого слова function, после которого идет символ \*. Внутри функции используется ключевое слово yield, которое указывает точки остановки во время выполнения генератора.

2) **Ленивое выполнение:** Генераторы работают по принципу ленивого выполнения. Это означает, что значения генерируются по требованию, только когда вызывается метод **next()** итератора. Таким образом, генераторы позволяют эффективно обрабатывать большие коллекции или бесконечные последовательности значений.

3) **Пауза и выполнение:** Ключевое слово **yield** используется для приостановки выполнения генератора и возврата значения. Когда вызывается метод **next()**, выполнение генератора возобновляется с точки приостановки и продолжает генерацию следующего значения.

4) **Итератор:** Генераторы автоматически предоставляют итератор, который можно использовать для последовательного получения значений. Итератор генератора реализует метод **next()**, который возвращает объект { value, done }. Свойство **value** содержит текущее сгенерированное значение, а свойство **done** указывает, завершен ли генератор.

1. Глобальный объект – особый объект, который является корневым объектом в глобальной области видимости. Он представляет собой контекст выполнения JavaScript-кода, который доступен в любой части программы.

В браузерной среде, глобальный объект называется window и предоставляет доступ к множеству встроенных функций и свойств, таких как console, alert, setTimeout, document и другие. Он также содержит глобальные переменные, объявленные без явного привязывания к какому-либо объекту.