**Каррирование (currying)**— преобразование функции от многих аргументов в набор функций, каждая из которых является функцией от одного аргумента.

Каррирование функции поможет нам избежать лишних телодвижений, если нам нужно использовать данную функцию в разных частях кода или вовсе в разных файлах.

Для примера возьмем обычную функцию сложения двух чисел и каррируем её.  
**Примечание:** Дочерняя функция всегда может использовать аргументы родительской функции, но **не** наоборот.

function sum(a, b) {

return a + b

}

function sumCurry(a) {

return (b) => {

return a + b

}

}

console.log(sumCurry(10)) //отобразит в консоли дочернюю функцию

В данном примере важно то, что при вызове функции **sumCurry(10)** значение аргумента **a** останется в памяти.

После таких нехитрых действий, мы можем вызывать данную функцию в любой момент и в ней всегда будет храниться значение.

const sumFunc = sumCurry(10)

console.log(sumFunc(5)) //отобразит в консоли 15

console.log(sumFunc(15)) //отобразит в консоли 25

Стоит понимать, что следующие записи будут равнозначны:

const sumFunc = sumCurry(10)

console.log(sumFunc(5)) //отобразит в консоли 15

console.log(sumCurry(10)(5)) //отобразит в консоли 15

**Еще пример каррирования**

const sendRequestCurrying = (host) => (baseRelativePath) => (url) => {

setTimeout(() => {

display.log(`${host}/${baseRelativePath}/${url}`)

}, 500)

}

const baseSendRequest = sendRequestCurrying('https://google.com')

const baseSendRequestWithPath = baseSendRequest('search')

baseSendRequestWithPath('vk')

baseSendRequestWithPath('facebook')

baseSendRequestWithPath('toyota')

baseSendRequestWithPath('gold')

<https://learn.javascript.ru/currying-partials>