**Currying in JavaScript**

1. **How Currying works**

Curried functions are those that **returns another function** (nested). This results in a **split of parameters**, and make the code looks neater, and create multiple functions for multiple purposes (instead of multi-purpose functions)

Moreover, it also brings about some supports to when we do not have enough arguments (yet) for all parameters.

For example, we can compare the 2 function calls.

|  |  |
| --- | --- |
| Normal Function | Curried Function |
| function add(a,b)  {     return a + b; } | function curried\_add(a) {     // The outer function returns a nested single-argument function     return function nested\_func(b) {         return a + b;    } } |
| let result = add(1,2) | let nested\_func = curried\_add(1);  let result = nested\_func (2)  OR  let result = curried\_add(1)(2) //curried\_add(1) returns the function |

*Note*: The argument from calling the outer function is available for the inner function due to something called *closure*.

A closure means that the nested function retains the scope of parent functions based on where the function is defined, even if the nested function is executed and finished outside of that lexical scope (based on the structure of the code).

1. **Currying with Arrow Functions**

The curried functions can also be written more precisely using arrow functions.

let curried\_add = a => b => a + b;

1. **Currying in Context**

For example, we are given an array of objects, each represents a player with some data (age, city, dateJoined, etc.)

Text

Description automatically generated

Now we would like to **filter** and **sort** the players, by various criteria. First we want to filter by city and sorted by age, then we want to filter by city and sport.

🡪 Each time we generate a list of filtered values, they have to repeat filters that iterate over the entire array. The codes are also hard to track, identify errors, and are unintuitive.

Text

Description automatically generated

Text

Description automatically generated

INSTEAD, with currying, we can guarantee the final filter will not execute until all arguments have been entered and each nested function has been called. Since you’re handling each argument one off in single argument functions, you can handle validating each argument as you go through the chain and hone in on any argument errors before code proceeds or has a chance to return an empty array.

We can create a curried function that filters an array of objects by a provided key and value.

Text

Description automatically generated

🡪 These codes are not only more modular, but also more reusable.

Now we can reuse the filtered San Francisco object for additional specialized functions. Let’s use it to sort the players by the date they joined the sports league.

Text

Description automatically generated