JavaScript Part 5: Arrays and Objects

Real business data comes in lists of records, such as orders, customers and invoices. JavaScript holds a list in an array and a record in an object. Four steps turn that data into answers: list, describe, transform and summarise. Each step answers one simple question, and the browser console is enough to try every line. In this part, Salma, a sales operations specialist at a regional distributor, uses them to report the month's orders.

1. List

An array is an ordered list written in square brackets, with the items separated by commas. Each item has a position called an index, and the first index is zero, not one. The length property tells us how many items the array holds, so the last item sits at the position that is one less than the length. The push method adds an item at the end. The question is: does the order matter here, and is this a list of similar things? Asking for a position that does not exist gives undefined, not an error. An array declared with const can still receive new items, because const only prevents the name from pointing to a different array.

Example: A queue at a service desk works like an array: the people stand in a fixed order, and the person at the front is position zero in a programmer's count, the next is position one, and so on. New arrivals join at the end of the line, exactly as the push method adds an item at the end of an array. If we ask how many people are waiting, the answer is the length of the array, and the last person always stands one position before that number.

At work: Salma receives the month's orders as a list. She asks the List question and keeps them in one array named orders, because the order of arrival matters for the report. A colleague adds late orders with push, and the length tells her how many orders she is dealing with. Nothing is copied into separate variables.

2. Describe

An object describes one thing with named properties, written as pairs of a name and a value inside curly braces. We read a property with a dot followed by its name, and we change a property or add a new one by assigning a value. The values can be strings, numbers, booleans, arrays or even other objects. The question is: what facts do we need to know about this thing? Real data is usually an array of objects, such as a list of orders in which every order has a customer, a region and an amount. An object declared with const can still have its properties changed, because const protects only the name. Using the same property names in every object makes the whole list easy to process.

Example: A box of business cards is a good picture of an array of objects, and each single card is one object. Every card has the same fields, a name, a phone number and a company, just as every order object has the same property names, and finding the phone number on any card is always the same action. In code, the dot followed by the property name is that action, so reading the phone number of one card is a single short expression, no matter which card we pick.

At work: Salma describes each order as an object with the properties customer, region, amount and paid. She asks the Describe question and checks that every order uses exactly the same property names. The paid property is a boolean, and the amount is a number rather than text. Reading the region of any customer is now a simple, readable expression.

3. Transform

The filter method creates a new array that contains only the items that pass a test. The map method creates a new array in which every item is replaced by the result of a function. Both receive a function, and the arrow functions from the previous part fit them perfectly. Neither of them changes the original array, which stays

This document is a property of Asaseeyat Academy - © Copy Right Protected.

available for other questions. The question is: which items do we keep, and what should each item become? For example, `orders.filter((order) => order.region === "North")` keeps only the orders of the north region. Chaining the two, first filter and then map, turns a long list of records into a short list of exactly the values we need.

Example: A recruiter has a pile of applications and first removes every one that does not meet the basic requirements, which is exactly what filter does, because it keeps only the items that pass a test. Then the recruiter writes down only the name and the phone number of each remaining person, which is what map does, because it turns every item into a new, smaller value. The original pile of applications is still on the desk, untouched, so if someone asks a different question, both filter and map can start again from the full pile.

At work: Salma filters the orders of the north region and maps them to their amounts. She asks the Transform question: which items do we keep, and what should each item become? The result is a clean list of amounts for that region, and the original orders array is untouched. She repeats the same pattern for the other regions with a different region name.

4. Summarise

The reduce method combines all the items of an array into a single value. It receives a function and a starting value, and in each round the function takes the running result and the current item. Adding up amounts is the classic use, with zero as the starting value. The find method returns the first item that passes a test, or undefined when none does. The question is: what single answer do we need from this list? For a question such as the total or the first unpaid order, one line with reduce or find replaces a whole loop. We must remember that find stops at the first match and says nothing about the others.

Example: A cashier closing the day does not read every receipt aloud, but adds them all into one running total that starts from zero, and that is what reduce does with the amounts of an array. Asked for the first receipt without a signature, the cashier checks the receipts one by one and stops the moment one is found, and that is what find does. So reduce turns the whole list into a single total, while find returns only the first match and says nothing about the receipts that come after it.

At work: Salma uses reduce on the north region's amounts, starting from zero, to produce the regional total. She asks the Summarise question and uses find to locate the first unpaid order, so that the finance team can call that customer first. The report that took an afternoon in a spreadsheet is produced in seconds. When a late order is added to the array, running the same code again updates every figure.

Conclusion

Salma's monthly report now comes from one array of orders and a few short lines. List asks whether the order matters and whether the items are alike, and describe asks what facts we need about each thing. Transform asks which items to keep and what each should become, and summarise asks what single answer the list must give. In the next part, we connect such data to a web page with the DOM and events.

Sources

- MDN Web Docs: Array – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
- MDN Web Docs: Working with objects – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_objects
- MDN Web Docs: Array.prototype.filter() – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter
- MDN Web Docs: Array.prototype.reduce() – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce
- MDN Web Docs: Array.prototype.find() – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/find
- MDN Web Docs: const – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const

This document is a property of Asaseeyat Academy - © Copy Right Protected.