**JavaScript Map, Filter and Reduce with simple examples -** [Constant Zomer](https://constantzomer.medium.com/?source=post_page-----25752a2e99b3--------------------------------) Feb 15, 2022

Map, Filter and Reduce are some of the most useful additions to JavaScript in version ES5. Let’s walk through a practical example of all three of them to teach you the basics of how to use them.

**Exploring distant planets**

Fizz Lightyear is an interplanetary explorer from planet Foo, a desert world in the galaxy of Bar.

We are programming his navigator that helps him keep track of which planets he goes to and for how long.

Let’s see his upcoming travel plans, an array of objects containing various data items:

A picture containing text, monitor, screen

Description automatically generated

As you can see part of Fizz’ time is spent back on his home planet Foo, so let’s use the filter() method to get rid of these entries.

**Filtering out the home planet**

The way filter() works is that it goes through our array and looks at every element of the array one by one. We can then give a rule for what passes the test and what doesn’t.

In this case, if the key ‘planet’ is equal to the value ‘Foo’ we should not let this pass the filter.



What you can see happening here is that we execute the filter() method on the planetVisits array. Within the filter we pass a function (in this case in the form of an ES6 arrow function) and sift through all the individual objects (‘planetObject’).

When the planet value of that planetObject is not equal to ‘Foo’ we return it as we want to keep it in the new array that we’re building . By returning it we will put that entire single planetObject into the ‘planetVisitsFiltered’ variable. What we’re left with is a new array of objects:

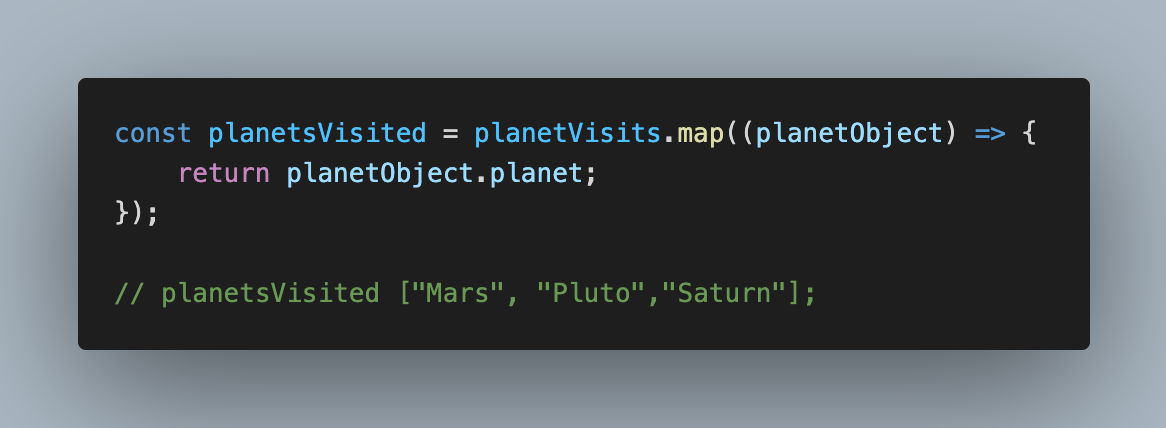
Text

Description automatically generated

**Mapping out the planets**

Now we’re asked to add functionality to the navigator that shows which planets he has visited (excluding his home planet).

What we can use for this is the map() method, this creates a new array by calling a function for every array element. What we should do is to only return the value of the key ‘planet’ from each planetObject:



Now there is just one last functionality we have to add: Fizz would like to know how many days he’ll spend on the planets in total.

**Reducing the days to a single number**

For this we can use the reduce() method. Reduce takes a set of numerical values and will return a single value from them. Reduce has 2 required arguments, the first is the ‘accumulator’ (or you could call it the sum) that keeps track of the total while going through the array values.

And then there is our planetObject which holds the amount of days Fizz spends on each planet. We take the amount of days from the planetObject and add it to the total which brings the value of ‘daysVisitingPlanets’ to 36. Instead of adding the number you can also use the other mathematical operators such as \* to multiply.

