**Array Cardio Day**

**.filter()**

Our first task it to get all the people who were born in the fifteen hundreds.

So we are going to make a variable that is const fifteen for fifteenhundreds and we are going to filter over it inventors.filter() and how you do this is pass it a function which is going to loop over every single item in our arrays and is going to give us our inventors and then we create an if statement which will decide if we keep it or not.

const fifteen **=** inventors.filter(function(inventors) {

so if on object inventor.year is greater than or equal to 1500 And the year is less than 1600 then return true.

const fifteenHundreds **=** inventors.filter(function(inventors) {

**if**(inventor.year **>=** 1500 **&&** inventor.year **<** 1600) {

**return** true; *// keep it*

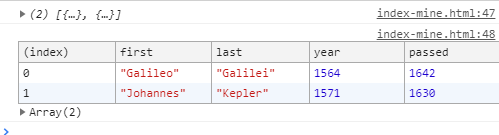
}

});

**console**.log(fifteenHundreds);

(neat trick) if you do console.table

**console**.table(fifteenHundreds);



We can reduce this to an arrow function by deleting the word function and removing the parentheses around inventors and put the arrow function behind inventors.

const fifteenHundreds **=** inventors.filter(inventors => {

and then also the if statement is either returning true or false so we can actually wright it inline to just return the two things in the if statement sense it will return a Boolean of true.

const fifteenHundreds **=** inventors.filter(inventors => (inventors.year **>=** 1500 **&&** inventors.year **<** 1600))

NEXT

**.map()**

Map take in an array, does something with it then returns a new array but ALSWAYS with the same length. So we do the object title inventors.map and then pass in inventors with a arrow function and concatenate inventors.first and inventors.last with a space in-between.

const fullnames **=** inventors.map(inventors => inventors.first **+** ' ' **+** inventors.last);

**console**.log(fullnames);

we can also do it by using template strings and back tic `${inventor.first}

const fullnames **=** inventors.map(inventors => `${inventors.first} ${inventors.last}`);

and this also works

**.sort()**

The way sort works is that you get two items and you are asked to sort just those two items. So its like asking is person A older than person B and if so you put the oldest person on top and the way we do that is by returning 1 and -1 and that is going to bubble theses items up and down in the array

const ordered **=** inventors.sort(function(firstPerson, secondPerson) {

you can also substitute (a , b) for (firstPerson, secondPerson)

then and if statement of a.year > b.year then return 1. If not (else) -1

and then a console.table(ordered);

const ordered **=** inventors.sort(function(firstPerson, secondPerson) {

**if**(firstPerson.year **>** secondPerson.year) {

**return** 1;

} **else** {

**return** **-**1;

}

});

**console**.table(ordered);

this can be condensed into a ternary operator which is like a short hand inline if statement

for arrow function (a, b) => ((if)) a.year is greater that b.year then ((if, then = ?)) 1 if not ((:)) -1

const ordered **=** inventors.sort((a, b) => a.year **>** b.year **?** 1 **:** **-**1);

**.reduce()**

What a reduce will do is allow you to build something on every single one

const totalYears **=** inventors.reduce()

and its supposed to give you your running total from last time/ what you have returned from this function last time. So we are going to pass it in a call it total and it also is going to give you the inventor (total, inventor)

const totalYears **=** inventors.reduce((total, inventor))

and from that we are going to return the total(which is what the total was last time around) and then return the number of years that inventor lived .

const totalYears **=** inventors.reduce((total, inventor) => {

**return** total **+** (inventor.passed **-** inventor.year);

});

**console**.log(totalYears);

but apparently It didn’t quite work because it gives us this object object



??? apparently you just put a zero on the end ???

const totalYears **=** inventors.reduce((total, inventor) => {

**return** total **+** (inventor.passed **-** inventor.year);

},0);

**console**.log(totalYears);



**.sort() another but with declared variables**

const oldest **=** inventors.sort(function(a, b){

we are going to do it just like before but this time we are going to declare some variables.

We are going to declare variables last guy and next guy and then the if statement of if lastGuy is greater than nextGuy then return -1 otherwise return 1

const oldest **=** inventors.sort(function(a, b){

const lastGuy **=** a.passed **-** a.year;

const nextGuy **=** b.passed **-** b.year;

**if** (lastGuy **>** nextGuy) {

**return** **-**1;

} **else** {

**return** 1;

}

});

And if we reduce it further with a inline ternary operator statement

*// if (lastGuy > nextGuy) {*

*// return -1;*

*// } else {*

*// return 1;*

*// }*

**return** lastGuy **>** nextGuy **?** **-**1 **:** 1;

and then console.table the oldest

const oldest **=** inventors.sort(function(a, b){

const lastGuy **=** a.passed **-** a.year;

const nextGuy **=** b.passed **-** b.year;

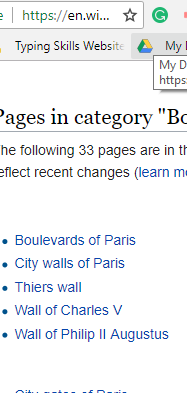
**return** lastGuy **>** nextGuy **?** **-**1 **:** 1;

});

**console**.table(oldest);

**next one is**

*create a list of Boulevards in Paris that contain 'de' anywhere in the name*

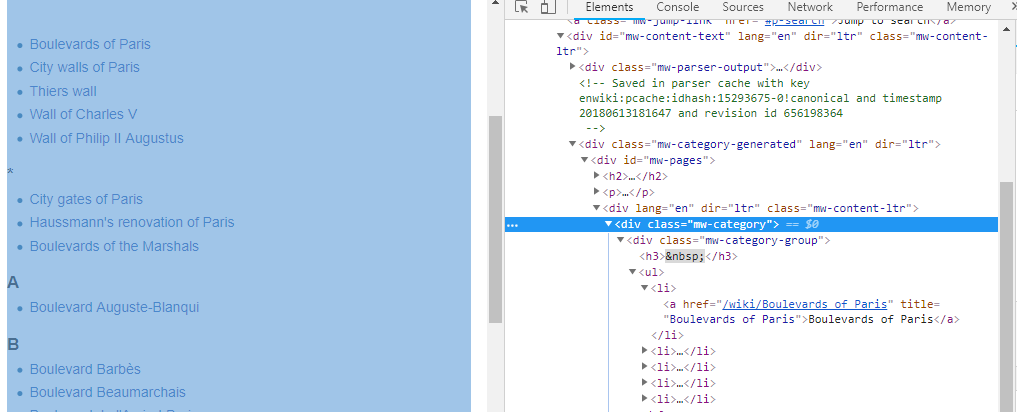


So we have in the wiki link given to us

<https://en.wikipedia.org/wiki/Category:Boulevards_in_Paris>

so we have all the boulevards on the page. We need to get the DOM elements out of the page.

So we could do document.queryselector but we don’t know what to select. So we have to get out our DEV Tools, inspect whatever contains this

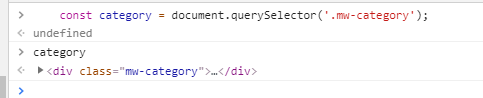


Which is <div class=”mw-category”>



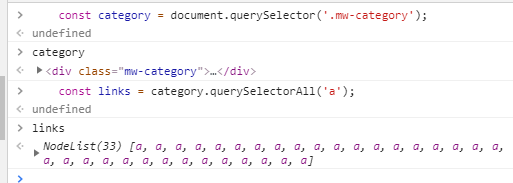
So

const category **=** **document**.querySelector('.mw-category');



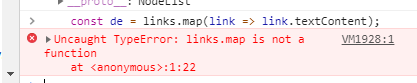
Then we want the links within it

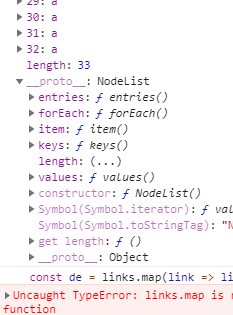
const links **=** category.querySelectorAll('a');



So now we want to convert this list of links to a list of names

const de **=** links.map(link => link.textContent);

uh ohh

Just like before in the earlier exercise query selector will return to you a node list and if we open up the prototype we don’t see map you see for each but not map

So what we need to do is to change this

const category **=** **document**.querySelector('.mw-category');

const links **=** category.querySelectorAll('a');

const de **=** links.map(link => link.textContent);

into an array. We could do it by just wrapping the entire thing in Array.from

const links **=** **Array**.from(category.querySelectorAll('a'));

or we can do it the ES6 way and create a array and use spead ((…))

const links **=** [**...**category.querySelectorAll('a')];

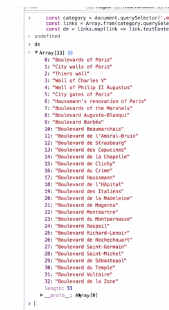
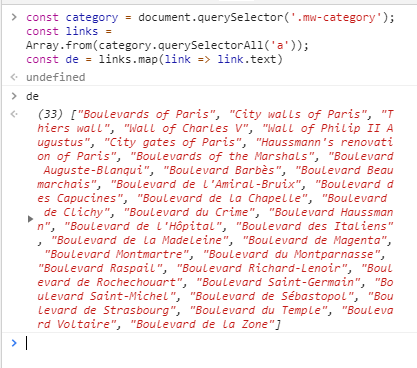
we will do it the first way though just a little but more readable

so

const category **=** **document**.querySelector('.mw-category');

const links **=** **Array**.from(category.querySelectorAll('a'));

const de **=** links.map(link => link.**text**)



So da is a huge array and what we are going to want to immediately do is filter that

So first we will map it and the next one we will filter it

const category **=** **document**.querySelector('.mw-category');

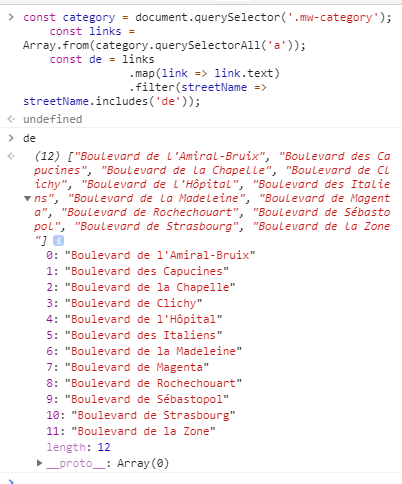
const links **=** **Array**.from(category.querySelectorAll('a'));

const de **=** links

.map(link => link.**text**)

.filter(streetName => streetName.includes('de'));

Now we have a subset of that array



**next one is a Sort exercise and we want to sort them by last name**

const alpha **=** people.sort(function(lastOne, nextOne){

**console**.log(lastOne);

})



But we want to convert that(lastOne) into a first name and a last name. so we add a

const parts **=** lastOne.split(', ')

since between the first and last name they are separated between a comma and a space

but now it’s a little bit backwards because the last name is first. What we can do is to de-structure that

const alpha **=** people.sort(function(lastOne, nextOne){

const [last, first] **=** lastOne.split(', ')

**console**.log(last, first);

so now we have to do that with the next one as well

const alpha **=** people.sort(function(lastOne, nextOne){

const [last, first] **=** lastOne.split(', ')

const [last, first] **=** nextOne.split(', ')

**console**.log(last, first);

*// const alpha = people.sort(function(lastOne, nextOne){*

And after condensing it down to a arrow function

const alpha **=** people.sort((lastOne, nextOne) => {

const [alast, afirst] **=** lastOne.split(', ')

const [blast, bfirst] **=** nextOne.split(', ')

**return** alast **>** blast **?** 1 **:** **-**1;

});

**console**.log(alpha);

**LAST one is reduce**

We want to tally up all the instances ((items)) in the array

const transportation **=** data.reduce(function(obj, item){

now we need to start with an object and set it initially to 0

const transportation **=** data.reduce(function(obj, item){

**if**(**!**obj[item]){

obj[item] **=** 0;

}

obj[item]**++**;

**return** obj;

}, {})

**console**.log(transportation);

so we start with a blank object and then every time we loop over one we first see if there is a zero there to work with at all and if not and we make a entry for that and then we go ahead and increment it.