2.4.1.4: Implementing a search function

Now we've got the beginnings of a little gallery app, so let's carry on developing it a little bit more. As you can see in this webpage here, I've added a little search box.

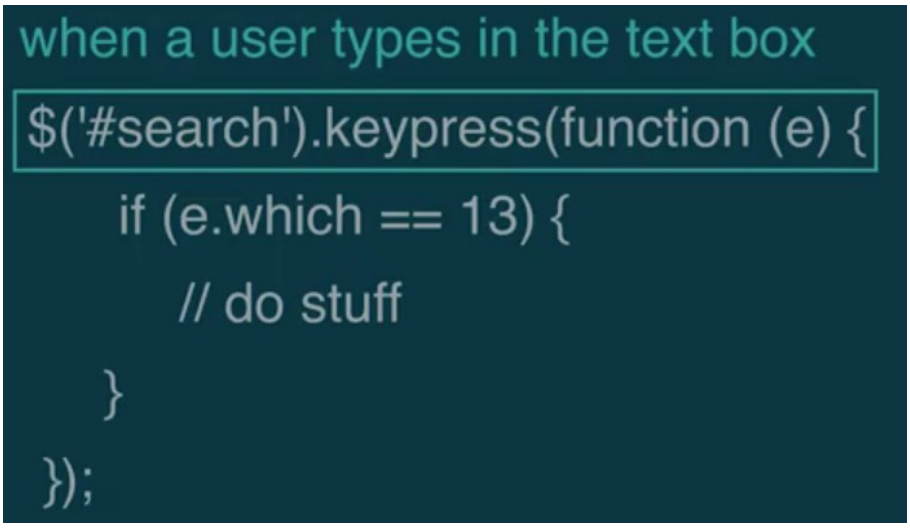
So, that's a really useful feature as you scale up to lots and lots and lots of images, or lots of blog posts, or whatever you have. It's very useful to be able to find it by searching. So, one way to do it is to have a search box here. If I type in horse, it'll find me the one with, the image that's called horse, there's only one, if I hit return with nothing in, it will go back to everything. If I do partial words, it'll find me a few that match. That's not really my house, by the way. So, this is a nice little bit of functionality. I'll go into a little bit more detail so I can explain it, exactly what I'm doing, I'm typing word, horse, and then hitting return, and the minute I hit return, it'll find stuff for me. And I'll show you how to implement that, step-by-step.

So, implementing that search process, as I showed you, involves three basic things:

- First, we've gotta recognize when somebody's actually searched for a word and figure out whatword they're searching for.
- Second, we've got to search through all our images in our array and find any images that are, that match that search word. And I'm gonna to do that. I'm gonna match both the title and the author.
- And finally, once you've got those images, we have to put them in a new array and re-display them, which is quite similar to how we displayed the original way.

So, let's take you through each of those steps in turn. The first thing we've got to do is figure out when somebody's actually typed some search text in there.

So, I'm going to use another jquery function, this time keypress, and that's just recognizes when somebody presses a key, types, and texts in the box.

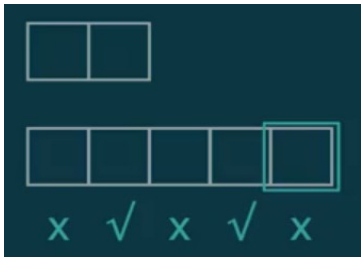


So, this happens every time somebody types an individual keyboard key into the search box. Now, sometimes it's good to be searching incrementally, and every single time a key is pressed, we start searching. In this case, I'm not actually doing that.

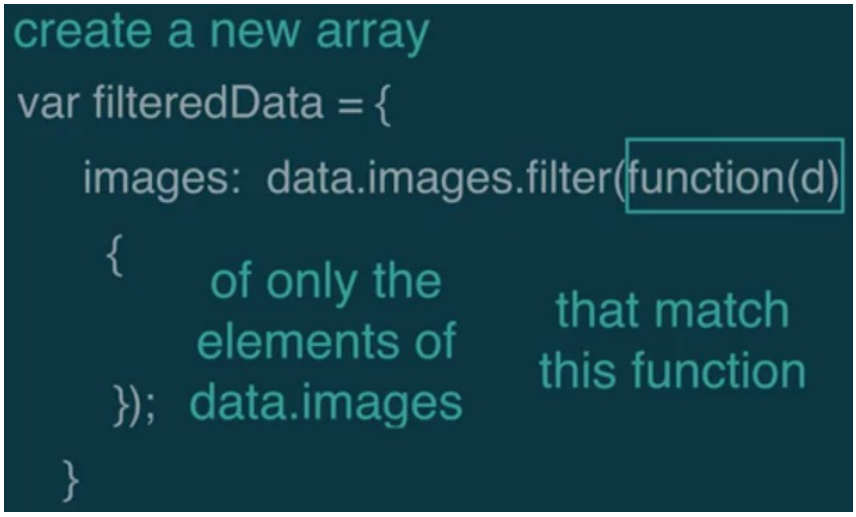
So, I'm waiting until the user hits return. And in order to spot that, I'm doing this line of code, which is checking exactly which key has been hit. So e is, this variable, **e.which** is actually the event variable, so it's passed into functions ever time we call an event, and it contains some useful data about what just happens. In this case, e.which tells you which is the exact key that has been pressed. Now, if we were looking at a standard alphabetic key, like a, b, c, this would be quite easy, we just need to look for if e.which is a, b, or c, but there's no real way of typing the return key other than the carrot and, sort of, new line, which is something different. So we can't actually find it based on typing in the character itself, so instead of that, all of the possible keyboard keys have a number associated with them as well, and in this case, we're gonna use the number and as you can see here, the number of the return key is 13.

Okay, so we found the, we've figured out whether we've hit the return key or not and whether we're typing something in. So we need to put something inside this statement to actually find all of the elements that match this search string.

To do that, we basically need to go through each element of the array, starting at number one, and checking, does it have the search string anywhere in it's author or title? No, leave it alone. Next one, yes, okay. We need to copy that next one into a new array and start filling out a new array with all the elements that do match the search string. Next one, no. Next one, yes. Excellent, another one for our array. Next one, no. So we've taken our original array and copied only those elements that matched the search string into a new array that we can now display.

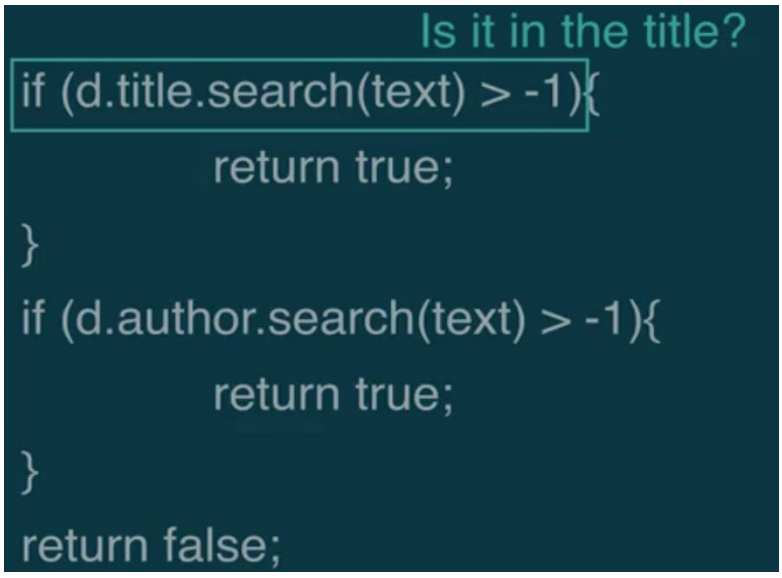


Let's go into detail intohow that searching works. So this is the exact function.



- So we're creating a new array, so we're creating a new variable. Now, just like our original variable, this is an object that contains a number of variable images, which is an array. As I said before, the reason we do that is so that we can pass it into that template, but the important thing is how we create this new array.
- And we're using this function filter, so data.images is our original array of images. So we're taking those and we're using the function filter. What is the function filtered to? It takes an array and looks through that array, finding only those elements that obey a certain property, so in this case, the contend search string. How do we tell it the property?
- We pass in another function, in this case, it's this anonymous function here, and what we're saying is we're finding only those elements of data.image that match this particular function. What this function has to do is that, one at a time it's given a member of the array, in this case an object, and it's got to tell you, does this contain the search string or does it not contain the search string. If it does contain the search string, it returns true, true means basically yes, it's a special value. If not, it returns false.

To do that, we have the special if statements.



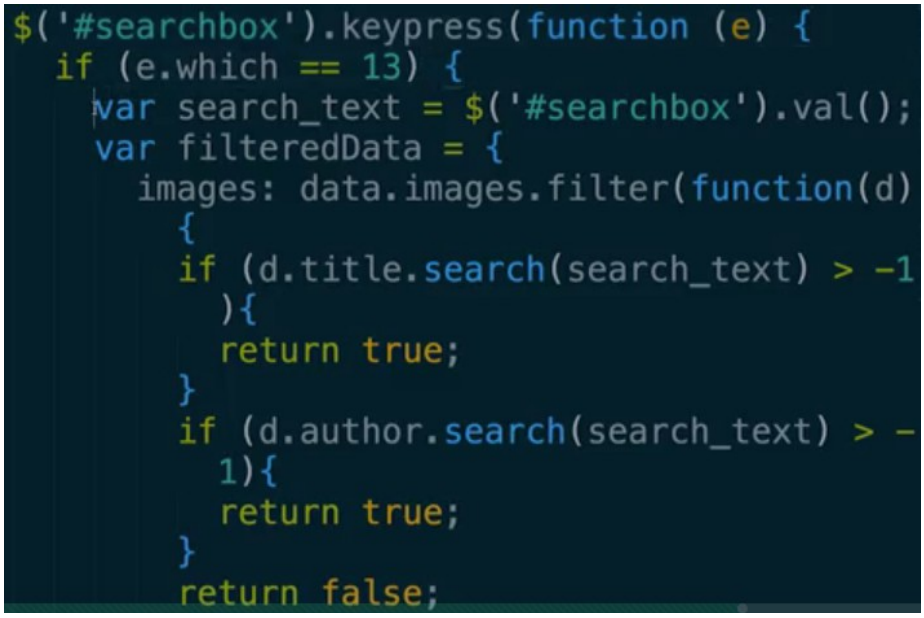
And the first one is saying, have we found the text inside the title? We're using d.title.search, which searches for a little bit of text inside a bigger bit of text, d.title is a bit of text, and we're searching inside that for the smaller bit of text called text. If it is, we return true, if not, we carry on.

Check the author, again if it's in there, return true, if not, carry on. And if we reach the end, we know we haven't found it, so we return false.

- So just before I leave that, I should explain this little bit, which is a bit strange. What's this saying? It's saying that if d.title.search is more than -1, why is that?

Well what d.title.search does is it searches through the big bit of text, d.title, for the small bit of text, text. And if it finds it, it returns the position in the text where the small text is. So, if the word horse is four characters into the title of the image, it will return three because the array starts at zero. So it's returning your number, which tells you the position of the search string within the bigger text. What happens if it's not found? Well, if it isn't found, it returns to -1, because there isn't any such position as -1, so it's a good number to say, yes we haven't found it, we'll just return a negative number. So going back to our case here, we can see that If you want to know where we found it, we don't care where it is. We at least know that it's got to be zero or more for it to be, to have been found in the title. If it's -1 or if it's negative, that means we definitely haven't found it, so we're just checking if it's more than -1, which means it's zero or more, and we found it somewhere in the text.

Here's the complete example codes, you can, you know, you don't have to take them all from video, we're gonna give you this complete example. You can download it but I just thought I'd show it quickly, just to see how everything fits together.



- So we've got this keypress function on the search box, we're checking if it's the carriage return, if it's the return key, then I didn't show you this bit,
- but we're just using searchbox.val, where that gets the value of the text that's been typed into the search box, it's another jquery function, and you can, resetting this variable search text like this.
- Then, we are showing this filtering code that I just showed you that is creating this new array, andwe are using inside our f-statements that search texts variable, to see if we've found that or not, returning true or false, and once we have done all that,

- we can just use this new filter data variable, so this is the equivalent of the original object, but with only those, only the objects that contain the search text in it. We run that through our template, so exactly the same code as before in our original codes to generate, but with different data, so not all of the data, and we run that through,
- generate some HTML, and put it into the content div, as usual that will generate us a new display with only those elements that match the original search text.

And we can just see that one more time. Let me do that again because that's a bug, in my case. Okay, and we can do that one more time just to see how this works. And typing that in will return us just the image that matches that.

So, we have been able to implement searching, a really useful feature and the way we've done searching is that where it is, it's not really anything to do with the structure of the web page or the HTML, it's all acting on the data, the content that's behind the scenes, so it's all working with arrays and objects and as you can, we'll see later, this is one of the benefits of having this structure content, the vision is that all of our more complex algorithms that we might want to run, searching, sorting, etc., can all work on the data. And then the code to actually display them, can all be the same, because I haven't really changed any of the HTML, any of the templates, or really any of the code that's substantiating the templates. I'm just passing in a different set of data, which is filtered from my search string.