**Ajax Type Ahead**

In this site/application we will be creating a type ahead feature. We want to be able to type in the search field and then bring up results from our given url. (Api)(cities json file)

First thing we need to do is to fetch the data then filter the array down from what has been typed in the box

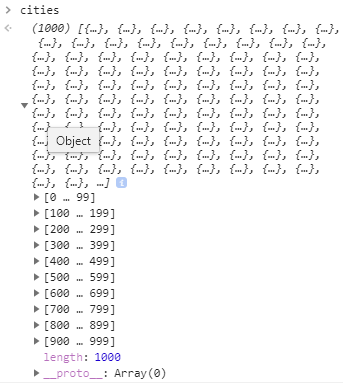
Here we turn the Jason into an array we can use

const cities **=** [];

fetch(endpoint)

.then(blob => blob.json())

.then(data => cities.push(**...**data));

Now we need to write a function that when I type in the search box it will filter the massive array down into a subset where you can then listen for it

function findMatches(wordToMatch, cities) {

**return** cities.filter(place => {

*//here we need to figure out if the city or state matches what was searched*

Now we can create a regular expression

**return** place.city.match(/wordToMatch/**i**)

but we want to put a variable into the wordToMatch statement

so we need to make a regular expression outside of it where we pass in the word to match (ex. bos, new York ) which is what we type into the search box and then we pass in any flags that we want. G is going to be global meaning It is going to look through the entire string for that specific one and I is going to be insensitive meaning that its going to match lower case as well

const regex **=** **RegExp**(wordToMatch, 'gi');

then we need to return the city or state

const regex **=** **RegExp**(wordToMatch, 'gi');

**return** place.city.match(regex) **||** place.state.match(regex)

and if this is true then it will filter our function

function findMatches(wordToMatch, cities) {

**return** cities.filter(place => {

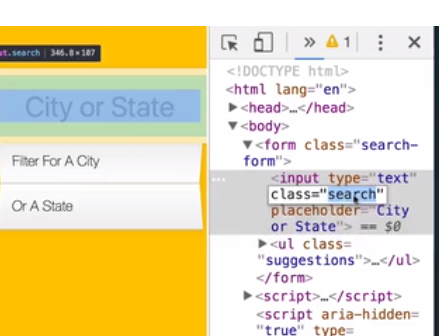
const regex **=** **new** RegExp(wordToMatch, 'gi');

**return** place.city.match(regex) **||** place.state.match(regex)

then we need to write our display function

function displayMatches()

and that is going to be called when we change the value in the search box



and we want to select this input that we are working with

const searchInput **=** **document**.querySelector('.search');

as well as suggestions



const searchInput **=** **document**.querySelector('.search');

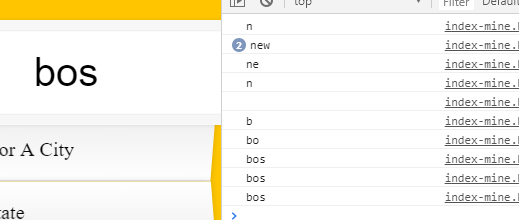
const suggestions **=** **document**.querySelector('.suggestions');

searchInput.addEventListener('change', displayMatches);

so now we are listening for the change event and when we put bos or new in the search box it will trigger this.value and when we click outside we can get a console log of the input but this is because the change event only fires when you go off that input so if we want to also have if fire when we type and key up we will duplicate this and we will listen for the key up event

searchInput.addEventListener('change', displayMatches);

searchInput.addEventListener('keyup', displayMatches);



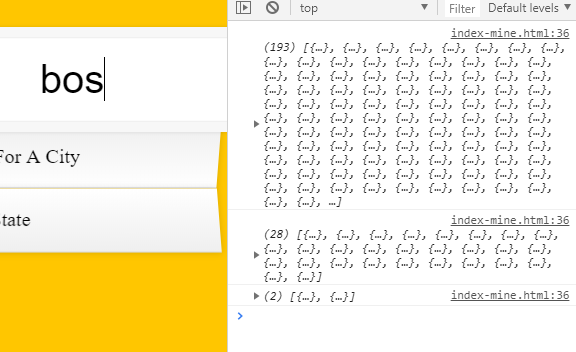
Now what we want to do is go back to the display matches function and we are going to use the find matches function.

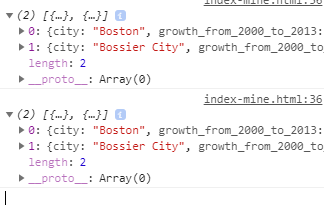
function displayMatches() {

const matchArray **=** findMatches(this.**value**, cities);

**console**.log(matchArray);

so whenever we type into it, it should run this, call our function and find one that matches it

so you can see when I typed B it returned 193 results, then when I typed o it returned 28 results and s to make bos I have 2 results that matched it

now WE HAVE THE DATA !

Best practice is to get the data first, get all your functionality in place and then we can worry about hooking it up to event listeners as well as hooking it up to the html that we need there.

So first we need to loop over it

const html **=** matchArray.map(place =>

from each of those we want to return

**return** `

<li>

<span class="name">${place.city}, ${place.state}</span>

<span class="name">${place.population}</span>

</li>

`;

And once we have mapped over this array

const html **=** matchArray.map(place => {

**return** `

<li>

<span class="name">${place.city}, ${place.state}</span>

<span class="name">${place.population}</span>

</li>

`;

We can simply call

suggestions.innerHTML **=** html;

however this map is going to return a array when what we want is just one string

so we can add a quick .join on the end and that will turn it into an array with multiple items into on big string

function displayMatches() {

const matchArray **=** findMatches(this.**value**, cities);

const html **=** matchArray.map(place => {

**return** `

<li>

<span class="name">${place.city}, ${place.state}</span>

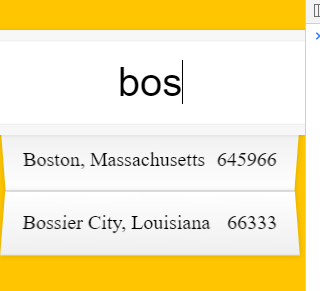
<span class="name">${place.population}</span>

</li>

`;

}).join('');

suggestions.innerHTML **=** html;

looking good. couple last things we need to do is format the numbers and highlight the actual words

So we go back to our map function here and before we return we can create a regex match the city name and use that regex to replace the word that it matches like WES with a span with a class hl and the word that it matches



const regex **=** **new** RegExp(this.**value**, 'gi');

const cityName **=** place.city.replace(regex, `<span class="hl">${this.**value**}</span>`)

so its going to find whatever it matched in the regex and replace it with a span class of highlight and then the value the person had searched for

so now we can replace cityName for place.city which works so duplicate it for state

function displayMatches() {

const matchArray **=** findMatches(this.**value**, cities);

const html **=** matchArray.map(place => {

const regex **=** **new** RegExp(this.**value**, 'gi');

const cityName **=** place.city.replace(regex, `<span class="hl">${this.**value**}</span>`)

const stateName **=** place.state.replace(regex, `<span class="hl">${this.**value**}</span>`)

**return** `

<li>

<span class="name">${cityName}, ${stateName}</span>

<span class="name">${place.population}</span>

</li>

`;

So finally what we need to do is find a function that is going to put in commas

This one is straight from stackoverflow

function numberWithCommas(x) {

**return** x.toString().replace(/**\B**(?=(\d**{3}**)**+**(?!\d))/**g**, ',');

}

So we need to update place.population with

numberWithCommas(place.population)

<li>

<span class="name">${cityName}, ${stateName}</span>

<span class="name">${numberWithCommas(place.population)}</span>

</li>

AND THAT’S IT!