(function(){ // protect the lemmings!

// init all the stuff

var checkboxesEl = document.querySelectorAll('input[type="checkbox"]');

var checkboxes = [].slice.call( checkboxesEl );

var idsAr = [];

var results = document.getElementById('results');

var searchValue = [0];

$(".filter").on ('click', function() {

searchValue[0] = ($(this).attr("data-value"));

//console.log (searchValue)

drawResults();

});

// ALL THE SORTS HERE

// list flags here

var flags = {

sortAlpha: 0

, sortNewest: 1

// ie: if sort location, put here

}

checkBoxClicks();

drawResults();

initAccordion();

// conditional stuff for the checkboxes

// DO NOT PUT AN ID FOR THE CHECKBOXES THAT DO THINGS LIKE

// SORT OR HIDE/SHOW STUFFS

document.getElementsByClassName('show-awards')[ 0 ].addEventListener('click', function() {

if ( this.checked ) {

document.getElementById('results').classList.add('show-awards-field');

}

else {

document.getElementById('results').classList.remove('show-awards-field');

}

});

document.getElementsByClassName('show-year')[ 0 ].addEventListener('click', function() {

if ( this.checked ) {

document.getElementById('results').classList.add('show-year-field');

}

else {

document.getElementById('results').classList.remove('show-year-field');

}

});

document.getElementsByClassName('sort-alpha')[ 0 ].addEventListener('click', function() {

if ( typeof idsAr === "undefined" ) return;

// make sure only one checkbox is clicked

toggleCheckbox('sort-alpha');

toggleFlag( this, 'sortAlpha' );

drawResults();

});

document.getElementsByClassName('sort-oldest-first')[ 0 ].addEventListener('click', function() {

if ( typeof idsAr === "undefined" ) return;

// make sure only one checkbox is clicked

toggleCheckbox('sort-oldest-first');

toggleFlag( this, 'sortNewest' );

drawResults();

});

// HANDLE CLICK EVENT FOR THE SORTING CHECKBOXES

function checkBoxClicks() {

checkboxes.forEach(function(el){

el.addEventListener('click', function() {

var isChecked = this.checked

, id = this.getAttribute('id');

if ( id === null ) return;

if ( isChecked ) {

idsAr.push( id );

}

else {

var idPos = idsAr.indexOf( id );

idsAr.splice( idPos, 1 );

}

drawResults();

});

});

}

// handle updatings results

function drawResults() {

if (searchValue % 2 === 0) {

var tags = getItemByTags\_ONLY( idsAr );

}

else {

var tags = getItemByTags\_ALL( idsAr );

}

results.innerHTML = "";

updateLabels();

if ( flags.sortAlpha ) {

tags = sortTags( tags );

}

else if ( flags.sortNewest ) {

tags = sortTagsByYear( tags );

}

else {

tags = sortTagsByYear( tags, true );

}

updateNumResults( tags.length );

tags.forEach(function(tag){

var div = document.createElement('div');

updateTitle( div, tag.title );

updateNotes( div, tag.notes );

updateTags( div, tag.tags );

updateYear( div, tag.year );

updateAwards( div, tag.awards );

results.appendChild( div );

});

} // drawResults

// sorts the tags alphabetically

function sortTags( tags ) {

tags.sort(function(a,b){

return ( a.title[ 0 ].toLowerCase() < b.title[ 0 ].toLowerCase() )

? -1

: 1;

});

return tags;

}

function sortTagsByYear( tags, oldFirst ) {

var newTags = [];

var tagsToSort = [];

var sortObj = {

first: 1

, second: -1

};

if ( oldFirst ) {

sortObj.first = -1;

sortObj.second = 1;

}

tags.forEach(function(tag){

if ( tag.year[ 0 ].toLowerCase() === 'o' ) {

newTags.push( tag );

}

else {

tagsToSort.push( tag );

}

});

tagsToSort.sort(function(a,b){

return ( parseInt( a.year, 10 ) <= parseInt( b.year, 10 ) )

? sortObj.first

: sortObj.second;

});

if ( oldFirst ) {

newTags = tagsToSort.concat( newTags );

}

else {

newTags = newTags.concat(tagsToSort);

}

return newTags;

}

// BEGIN DOM UPDATES HERE

// THESE FUNCTIONS HELP UPDATE THE RESULTS VIEW

function updateNumResults( numResults ) {

document.getElementById('num-results').innerHTML = numResults + " results";

}

function updateLabels() {

var labels = document.getElementById('labels');

labels.innerHTML = "";

for( var i = 0; i < idsAr.length; ++i ) {

var span = document.createElement('span');

span.classList.add('tag-name');

span.innerHTML = idsAr[ i ];

labels.appendChild( span );

}

} // updateLabels

function updateTitle( div, title ) {

var h2 = document.createElement('h2');

h2.innerHTML = title;

div.appendChild( h2 );

}

function updateNotes( div, notes ) {

var span = document.createElement('span');

span.innerHTML = notes;

span.classList.add('notes');

div.appendChild( span );

} // updateNotes

function updateTags( div, tags ) {

var ul = document.createElement('ul');//make an element in html called ul that covers the tags for each function

tags.forEach(function(currentTag){

var li = document.createElement('li');//make an element in html called

li.innerHTML = currentTag;//li is made out of the current tags

ul.appendChild( li );

});

div.appendChild( ul );

} // updateTags

function updateAwards( div, awards ) {

if (typeof awards !== "undefined"){

var h4 = document.createElement('h4');

h4.innerHTML = awards;

h4.classList.add ('awards');

div.appendChild( h4 );

}

} // updateAwards

function updateYear( div, year ) {

if (typeof year !== "undefined"){

var h5 = document.createElement('h5');

h5.innerHTML = year;

h5.classList.add ('year');

div.appendChild( h5 );

}

}

function toggleFlag( el, varToToggle ) {

// update flags

flags[varToToggle] = !flags[ varToToggle ];

}

function toggleCheckbox( classToExclude ) {

[].slice.call(document.getElementsByClassName('sort')).forEach(function(el){

if ( el.classList.contains( classToExclude ) ) {

return true;

}

else {

if ( el.checked ) {

el.checked = false;

toggleFlag( el, el.getAttribute('data-flag') );

}

}

});

}

//accordian

function initAccordion() {

$('.section h2').on('click', function() {

$( this )

.parent()

.find('.content')

.slideToggle({

duration: 'fast'

, easing: 'linear'

});

var caretClass, caretClassNew;

if ( $(this).find('.fa-caret-right').length !== 0 ) {

caretClass = 'fa-caret-right';

caretClassNew = 'fa-caret-down';

}

else {

caretClass = 'fa-caret-down';

caretClassNew = 'fa-caret-right';

}

$(this)

.find('.'+caretClass)

.removeClass(caretClass)

.addClass(caretClassNew);

});

}

// main search code is here

function getItemByTags\_ALL( tagsArray ) {

var searchResults = [];//WHERE THEY GET PUSHED TO

console.log (projects);

for( var i = 0; i < projects.length; ++i ) {//SEARCHES PROJECT ARRAY

var currentProject = projects[ i ];//THE PROJ IN THE ITH POSITION

var currentTags = currentProject.tags;//THE PROJ IN THE ITH POSITION'STAG

var tagsAsObj = {};

for ( var j = 0; j < currentTags.length; j++ ) {//SEARCHES THE PROJ IN THE ITH POSITION'STAG

tagsAsObj[ currentTags[ j ] ] = 1;// TAO is THE JTH PLACE OF

//THE PROJ IN THE ITH POSITION'STAG = 1

//tagsArObj[ "fourstar" ]

// tagsAsObj.fourstar = 1

} // j

//console.log( tagsAsObj );//LOGS THE TAGS ARRAY

for( var k = 0; k < tagsArray.length; k++ ) { //GO THROUGH THE JTH PLACE(THE PROJECT IN THE ITH PLACE"S TAG)

if ( tagsAsObj[ tagsArray[ k ] ] !== 1 ) {//IF IT EQUALS ONE THE VALE OF THE TAGS AS OBJECTS VAR

// searchResults.push( currentProject ); //PUSH THAT PROJ TO SEARCH OBJECTS

searchResults.push( currentProject );

}

}

}

return searchResults;

} // getItemByTags\_ALL

function getItemByTags\_ONLY( tagsArray ) {

var searchResults = [];//WHERE THEY GET PUSHED TO

for( var i = 0; i < projects.length; ++i ) {//SEARCHES PROJECT ARRAY

var currentProject = projects[ i ];//THE PROJ IN THE ITH POSITION

var currentTags = currentProject.tags;//THE PROJ IN THE ITH POSITION'STAG

var tagsAsObj = {};

for ( var j = 0; j < currentTags.length; j++ ) {//SEARCHES THE PROJ IN THE ITH POSITION'STAG

tagsAsObj[ currentTags[ j ] ] = 1;// TAO is THE JTH PLACE OF

//THE PROJ IN THE ITH POSITION'STAG = 1

//tagsArObj[ "fourstar" ]

// tagsAsObj.fourstar = 1

} // j

//console.log( tagsAsObj );//LOGS THE TAGS ARRAY

var hasAll = true;

for( var k = 0; k < tagsArray.length; k++ ) { //GO THROUGH THE JTH PLACE(THE PROJECT IN THE ITH PLACE"S TAG)

if ( tagsAsObj[ tagsArray[ k ] ] !== 1 ) {//IF IT EQUALS ONE THE VALE OF THE TAGS AS OBJECTS VAR

// searchResults.push( currentProject ); //PUSH THAT PROJ TO SEARCH OBJECTS

hasAll = false;

}

}

if ( hasAll ) {

searchResults.push( currentProject );

}

}

return searchResults;

} // getItemByTags

})();