Website-JS&css

Mario Niemand

MCSD

CTU-2018

Table of Contents

[Login Page: 3](#_Toc5203070)

[Sorting Page: 4](#_Toc5203071)

[Carousel Page: 4](#_Toc5203072)

[Code for login.html: 5](#_Toc5203073)

[Code for Sorting page: 5](#_Toc5203074)

[Code for Carousel: 6](#_Toc5203075)

[External CSS: 7](#_Toc5203076)

[External JavaScript: 13](#_Toc5203077)

# Login Page:

At this login page I used a simple design but the users is stored in a xml file.

# Sorting Page:

Here I used a sorting array that creates random data then sorts them or shuffles them.

# Carousel Page:

Here I created a advanced carousel that scrolls through images.

# Code for login.html:

<!DOCTYPE html>

<html>

<head>

<title>Login page</title>

<script src="JavaScript.js"></script>

<script src="jquery-3.3.1.min.js" ;></script>

</head>

<body>

<div align="right">

<h1 style="font-family:Times New Roman;text-align="center" ;font-size:20pt;

color:#00FF00;>

Welcome to Login Page

</h1>

<form name="login">

Username: &nbsp <input type="text" id="userID" /><br><br> <!--field wat in xml gan soek-->

Password: &nbsp <input type="password" id="password" /><br><br>

<input type="button" onclick="login2();" value="Login" id="btnLogin" /> <!--id na login function-->

<input type="reset" value="Cancel" />

</form>

</div>

</body>

</html>

# Code for Sorting page:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title></title>

<link rel="stylesheet" href="StyleSheet.css"><!--link to css-->

<meta charset="UTF-8">

<meta name="description" content="My FA1">

<meta name="keywords" content="MarioNiemand,FA2,HTML,CSS,JavaScript">

<meta name="author" content="Mario Niemand">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<script src="JavaScript.js"></script><!--link to javascript-->

<script src="https://cdn.rawgit.com/elving/wait/master/wait.js"></script><!--link to jquery-->

</head>

<body>

<canvas id="myCanvas"></canvas>

<br />

<input id="animSpeed" value="-33" type="range" min="-33" max="-1">

<br />

<div id="wrapper">

<span id="slow">slow</span> <!--here the user chooses the sorting speed-->

<span id="fast">fast</span>

</div>

<div id="controlPanel">

<button id="shuffleArray">Shuffle Array</button> <!--Shuffle button-->

<button id="bubbleSort">Bubble Sort</button> <!--sort button-->

<select id="selectOrder">

<option value="ascending">Ascending Order</option> <!--here the user can choose ascending or descending order-->

<option value="descending">Descending Order</option>

</select>

<br />

<div id="info"></div>

<br />

<div id="debug"></div>

<br />

</div>

</body>

</html>

# Code for Carousel:

<!DOCTYPE html>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>3d carousel 3</title>

<script src="JavaScript.js"></script>

<script src="jquery-3.3.1.min.js" ;></script>

<link rel="stylesheet" href="StyleSheet.css">

</head>

<body>

<style>

.carousel {

margin: 0 auto;

}

</style>

<div class="carousel">

<img src="images/img1.jpg" width="50%" height="50%" align="middle" />

<img src="images/img4.jpg" alt="Image 4" width="50%" height="50%" align="middle" />

<img src="images/img5.jpg" alt="Image 5" width="50%" height="50%" align="middle" />

</div>

<script src='http://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js'></script>

<script src="js/index.js"></script>

<br>

<br>

<br>

<br>

<br>

<br><p align="center">

Coding 101

MCSD course

2018

Mario Niemand

Carousel Page

</p>

</body>

</html>

# External CSS:

body {

background-image: url("images/bg2.jpg");

background-position: center;

}

.carousel {

width: 800px;

height: 300px;

display: relative;

}

.carousel img {

display: hidden; /\* hide images until carousel prepares them \*/

cursor: pointer; /\* not needed if you wrap carousel items in links \*/

}

#myCanvas {

border: solid 1px;

}

#wrapper {

background: red;

overflow: auto;

font-family: "Times New Roman";

}

#myCanvas {

border: solid 1px;

}

#wrapper {

background: red;

overflow: auto;

font-family: "Times New Roman";

}

#slow {

float: left;

}

#fast {

float: right;

}

#algorithmDetails {

font-family: "Times New Roman";

border: solid 1px;

}

#animSpeed {

width: 100%;

}

#slow {

float: left;

}

#fast {

float: right;

}

#algorithmDetails {

font-family: "Times New Roman";

border: solid 1px;

}

#animSpeed {

width: 100%;

}

form {

border: 0px solid #f1f1f1;

}

input[type=text], input[type=password] {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

display: inline-block;

border: 1px solid #ccc;

box-sizing: border-box;

}

button {

background-color: #4CAF50;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 100%;

}

button:hover {

opacity: 0.8;

}

.cancelbtn {

width: auto;

padding: 10px 18px;

background-color: #f44336;

}

.imgcontainer {

text-align: center;

margin: 24px 0 12px 0;

}

.modal {

display: none; /\* Hidden by default \*/

position: fixed; /\* Stay in place \*/

z-index: 1; /\* Sit on top \*/

left: 0;

top: 0;

width: 100%; /\* Full width \*/

height: 100%; /\* Full height \*/

overflow: auto; /\* Enable scroll if needed \*/

background-color: rgb(0,0,0); /\* Fallback color \*/

background-color: rgba(0,0,0,0.4); /\* Black w/ opacity \*/

padding-top: 60px;

}

/\* Modal Content/Box \*/

.modal-content {

background-color: #fefefe;

margin: 5px auto; /\* 15% from the top and centered \*/

border: 1px solid #888;

width: 80%; /\* Could be more or less, depending on screen size \*/

}

/\* The Close Button \*/

.close {

/\* Position it in the top right corner outside of the modal \*/

position: absolute;

right: 25px;

top: 0;

color: #000;

font-size: 35px;

font-weight: bold;

}

/\* Close button on hover \*/

.close:hover,

.close:focus {

color: red;

cursor: pointer;

}

/\* Add Zoom Animation \*/

.animate {

-webkit-animation: animatezoom 0.6s;

animation: animatezoom 0.6s

}

@-webkit-keyframes animatezoom {

from {

-webkit-transform: scale(0)

}

to {

-webkit-transform: scale(1)

}

}

@keyframes animatezoom {

from {

transform: scale(0)

}

to {

transform: scale(1)

}

}

#check {

border-radius: 75px;

}

img.avatar {

width: 40%;

border-radius: 50%;

}

.container {

padding: 16px;

}

span.psw {

float: right;

padding-top: 16px;

}

/\* Change styles for span and cancel button on extra small screens \*/

@media screen and (max-width: 300px) {

span.psw {

display: block;

float: none;

}

.cancelbtn {

width: 100%;

}

}

h1 /\*edits all headers \*/ {

font-family: Ubuntu;

font-size: 40px;

font-style: normal;

font-variant: normal;

font-weight: 500;

line-height: 26.4px;

}

#btnTop /\*java button \*/ {

display: none;

position: fixed;

bottom: 20px;

right: 30px;

z-index: 99;

border: none;

outline: none;

Background-color: blue;

color: white;

cursor: pointer;

padding: 15px;

border-radius: 10px;

font-size: 18px;

}

#btnTop:hover {

background-color: #555

}

#hireME {

float: right;

margin-right: 500;

}

#media /\*img \*/ {

float: right;

margin-top: -100;

}

#battle {

float: right;

}

#battlefield {

float: left;

}

#video /\*video css \*/ {

margin-left: 100;

margin-top: -60;

}

#container {

width: 400px;

height: 400px;

position: relative;

background: yellow;

}

#animate {

width: 50px;

height: 50px;

position: absolute;

background: red;

}

#game {

margin-left: 265;

margin-top: -30;

}

#stick {

float: right;

margin-top: -450;

}

.fa {

padding: 20px;

font-size: 30px;

width: 30px;

text-align: center;

text-decoration: none;

margin: 5px 2px;

border-radius: 50%;

}

.fa:hover {

opacity: 0.7;

}

.fa-facebook {

background: #3B5998;

color: white;

}

.fa-twitter {

background: #55ACEE;

color: white;

}

.fa-google {

background: #dd4b39;

color: white;

}

.fa-linkedin {

background: #007bb5;

color: white;

}

.fa-youtube {

background: #bb0000;

color: white;

}

.fa-instagram {

background: #125688;

color: white;

}

.fa-pinterest {

background: #cb2027;

color: white;

}

.fa-snapchat-ghost {

background: #fffc00;

color: white;

text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;

}

.fa-skype {

background: #00aff0;

color: white;

}

.fa-android {

background: #a4c639;

color: white;

}

.fa-dribbble {

background: #ea4c89;

color: white;

}

.fa-vimeo {

background: #45bbff;

color: white;

}

.fa-tumblr {

background: #2c4762;

color: white;

}

.fa-vine {

background: #00b489;

color: white;

}

.fa-foursquare {

background: #45bbff;

color: white;

}

.fa-stumbleupon {

background: #eb4924;

color: white;

}

.fa-flickr {

background: #f40083;

color: white;

}

.fa-yahoo {

background: #430297;

color: white;

}

.fa-soundcloud {

background: #ff5500;

color: white;

}

.fa-reddit {

background: #ff5700;

color: white;

}

.fa-rss {

background: #ff6600;

color: white;

}

# External JavaScript:

window.onscroll = function () { scrollFunction(); }

function scrollFunction() {

if (document.body.scrollTop > 20 || document.documentElement.scrollTop > 20) /\*checks if user has scrolled down to make button appear \*/ { document.getElementById("btnTop").style.display = "block"; }

else {

document.getElementById("btnTop").style.display = "none";

}

}

function topFunction() {

document.body.scrollTop = 0;

document.documentElement.scrollTop = 0; //back to top button

}

; (function ($) {

'use strict';

$.fn.waterwheelCarousel = function (startingOptions) {

// Adds support for intializing multiple carousels from the same selector group

if (this.length > 1) {

this.each(function () {

$(this).waterwheelCarousel(startingOptions);

});

return this; // allow chaining

}

var carousel = this;

var options = {};

var data = {};

function initializeCarouselData() {

data = {

itemsContainer: $(carousel),

totalItems: $(carousel).find('img').length,

containerWidth: $(carousel).width(),

containerHeight: $(carousel).height(),

currentCenterItem: null,

previousCenterItem: null,

items: [],

calculations: [],

carouselRotationsLeft: 0,

currentlyMoving: false,

itemsAnimating: 0,

currentSpeed: options.speed,

intervalTimer: null,

currentDirection: 'forward',

leftItemsCount: 0,

rightItemsCount: 0,

performingSetup: true

};

data.itemsContainer.find('img').removeClass(options.activeClassName);

}

/\*\*

\* This function will set the autoplay for the carousel to

\* automatically rotate it given the time in the options

\* Can clear the autoplay by passing in true

\*/

function autoPlay(stop) {

// clear timer

clearTimeout(data.autoPlayTimer);

// as long as no stop command, and autoplay isn't zeroed...

if (!stop && options.autoPlay !== 0) {

// set timer...

data.autoPlayTimer = setTimeout(function () {

// to move the carousl in either direction...

if (options.autoPlay > 0) {

moveOnce('forward');

} else {

moveOnce('backward');

}

}, Math.abs(options.autoPlay));

}

}

/\*\*

\* This function will preload all the images in the carousel before

\* calling the passed in callback function. This is only used so we can

\* properly determine the width and height of the items. This is not needed

\* if a user instead manually specifies that information.

\*/

function preload(callback) {

if (options.preloadImages === false) {

callback();

return;

}

var $imageElements = data.itemsContainer.find('img'), loadedImages = 0, totalImages = $imageElements.length;

$imageElements.each(function () {

$(this).bind('load', function () {

// Add to number of images loaded and see if they are all done yet

loadedImages += 1;

if (loadedImages === totalImages) {

// All done, perform callback

callback();

return;

}

});

// May need to manually reset the src to get the load event to fire

// http://stackoverflow.com/questions/7137737/ie9-problems-with-jquery-load-event-not-firing

$(this).attr('src', $(this).attr('src'));

// If browser has cached the images, it may not call trigger a load. Detect this and do it ourselves

if (this.complete) {

$(this).trigger('load');

}

});

}

/\*\*

\* Makes a record of the original width and height of all the items in the carousel.

\* If we re-intialize the carousel, these values can be used to re-establish their

\* original dimensions.

\*/

function setOriginalItemDimensions() {

data.itemsContainer.find('img').each(function () {

if ($(this).data('original\_width') == undefined || options.forcedImageWidth > 0) {

$(this).data('original\_width', $(this).width());

}

if ($(this).data('original\_height') == undefined || options.forcedImageHeight > 0) {

$(this).data('original\_height', $(this).height());

}

});

}

/\*\*

\* Users can pass in a specific width and height that should be applied to every image.

\* While this option can be used in conjunction with the image preloader, the intended

\* use case is for when the preloader is turned off and the images don't have defined

\* dimensions in CSS. The carousel needs dimensions one way or another to work properly.

\*/

function forceImageDimensionsIfEnabled() {

if (options.forcedImageWidth && options.forcedImageHeight) {

data.itemsContainer.find('img').each(function () {

$(this).width(options.forcedImageWidth);

$(this).height(options.forcedImageHeight);

});

}

}

/\*\*

\* For each "visible" item slot (# of flanking items plus the middle),

\* we pre-calculate all of the properties that the item should possess while

\* occupying that slot. This saves us some time during the actual animation.

\*/

function preCalculatePositionProperties() {

// The 0 index is the center item in the carousel

var $firstItem = data.itemsContainer.find('img:first');

data.calculations[0] = {

distance: 0,

offset: 0,

opacity: 1

}

// Then, for each number of flanking items (plus one more, see below), we

// perform the calcations based on our user options

var horizonOffset = options.horizonOffset;

var separation = options.separation;

for (var i = 1; i <= options.flankingItems + 2; i++) {

if (i > 1) {

horizonOffset \*= options.horizonOffsetMultiplier;

separation \*= options.separationMultiplier;

}

data.calculations[i] = {

distance: data.calculations[i - 1].distance + separation,

offset: data.calculations[i - 1].offset + horizonOffset,

opacity: data.calculations[i - 1].opacity \* options.opacityMultiplier

}

}

// We performed 1 extra set of calculations above so that the items that

// are moving out of sight (based on # of flanking items) gracefully animate there

// However, we need them to animate to hidden, so we set the opacity to 0 for

// that last item

if (options.edgeFadeEnabled) {

data.calculations[options.flankingItems + 1].opacity = 0;

} else {

data.calculations[options.flankingItems + 1] = {

distance: 0,

offset: 0,

opacity: 0

}

}

}

/\*\*

\* Here we prep the carousel and its items, like setting default CSS

\* attributes. All items start in the middle position by default

\* and will "fan out" from there during the first animation

\*/

function setupCarousel() {

// Fill in a data array with jQuery objects of all the images

data.items = data.itemsContainer.find('img');

for (var i = 0; i < data.totalItems; i++) {

data.items[i] = $(data.items[i]);

}

// May need to set the horizon if it was set to auto

if (options.horizon === 0) {

if (options.orientation === 'horizontal') {

options.horizon = data.containerHeight / 2;

} else {

options.horizon = data.containerWidth / 2;

}

}

// Default all the items to the center position

data.itemsContainer

.css('position', 'relative')

.find('img')

.each(function () {

// Figure out where the top and left positions for center should be

var centerPosLeft, centerPosTop;

if (options.orientation === 'horizontal') {

centerPosLeft = (data.containerWidth / 2) - ($(this).data('original\_width') / 2);

centerPosTop = options.horizon - ($(this).data('original\_height') / 2);

} else {

centerPosLeft = options.horizon - ($(this).data('original\_width') / 2);

centerPosTop = (data.containerHeight / 2) - ($(this).data('original\_height') / 2);

}

$(this)

// Apply positioning and layering to the images

.css({

'left': centerPosLeft,

'top': centerPosTop,

'visibility': 'visible',

'position': 'absolute',

'z-index': 0,

'opacity': 0

})

// Give each image a data object so it remembers specific data about

// it's original form

.data({

top: centerPosTop,

left: centerPosLeft,

oldPosition: 0,

currentPosition: 0,

depth: 0,

opacity: 0

})

// The image has been setup... Now we can show it

.show();

});

}

/\*\*

\* All the items to the left and right of the center item need to be

\* animated to their starting positions. This function will

\* figure out what items go where and will animate them there

\*/

function setupStarterRotation() {

options.startingItem = (options.startingItem === 0) ? Math.round(data.totalItems / 2) : options.startingItem;

data.rightItemsCount = Math.ceil((data.totalItems - 1) / 2);

data.leftItemsCount = Math.floor((data.totalItems - 1) / 2);

// We are in effect rotating the carousel, so we need to set that

data.carouselRotationsLeft = 1;

// Center item

moveItem(data.items[options.startingItem - 1], 0);

data.items[options.startingItem - 1].css('opacity', 1);

// All the items to the right of center

var itemIndex = options.startingItem - 1;

for (var pos = 1; pos <= data.rightItemsCount; pos++) {

(itemIndex < data.totalItems - 1) ? itemIndex += 1 : itemIndex = 0;

data.items[itemIndex].css('opacity', 1);

moveItem(data.items[itemIndex], pos);

}

// All items to left of center

var itemIndex = options.startingItem - 1;

for (var pos = -1; pos >= data.leftItemsCount \* -1; pos--) {

(itemIndex > 0) ? itemIndex -= 1 : itemIndex = data.totalItems - 1;

data.items[itemIndex].css('opacity', 1);

moveItem(data.items[itemIndex], pos);

}

}

/\*\*

\* Given the item and position, this function will calculate the new data

\* for the item. One the calculations are done, it will store that data in

\* the items data object

\*/

function performCalculations($item, newPosition) {

var newDistanceFromCenter = Math.abs(newPosition);

// Distance to the center

if (newDistanceFromCenter < options.flankingItems + 1) {

var calculations = data.calculations[newDistanceFromCenter];

} else {

var calculations = data.calculations[options.flankingItems + 1];

}

var distanceFactor = Math.pow(options.sizeMultiplier, newDistanceFromCenter)

var newWidth = distanceFactor \* $item.data('original\_width');

var newHeight = distanceFactor \* $item.data('original\_height');

var widthDifference = Math.abs($item.width() - newWidth);

var heightDifference = Math.abs($item.height() - newHeight);

var newOffset = calculations.offset

var newDistance = calculations.distance;

if (newPosition < 0) {

newDistance \*= -1;

}

if (options.orientation == 'horizontal') {

var center = data.containerWidth / 2;

var newLeft = center + newDistance - (newWidth / 2);

var newTop = options.horizon - newOffset - (newHeight / 2);

} else {

var center = data.containerHeight / 2;

var newLeft = options.horizon - newOffset - (newWidth / 2);

var newTop = center + newDistance - (newHeight / 2);

}

var newOpacity;

if (newPosition === 0) {

newOpacity = 1;

} else {

newOpacity = calculations.opacity;

}

// Depth will be reverse distance from center

var newDepth = options.flankingItems + 2 - newDistanceFromCenter;

$item.data('width', newWidth);

$item.data('height', newHeight);

$item.data('top', newTop);

$item.data('left', newLeft);

$item.data('oldPosition', $item.data('currentPosition'));

$item.data('depth', newDepth);

$item.data('opacity', newOpacity);

}

function moveItem($item, newPosition) {

// Only want to physically move the item if it is within the boundaries

// or in the first position just outside either boundary

if (Math.abs(newPosition) <= options.flankingItems + 1) {

performCalculations($item, newPosition);

data.itemsAnimating++;

$item

.css('z-index', $item.data().depth)

// Animate the items to their new position values

.animate({

left: $item.data().left,

width: $item.data().width,

height: $item.data().height,

top: $item.data().top,

opacity: $item.data().opacity

}, data.currentSpeed, options.animationEasing, function () {

// Animation for the item has completed, call method

itemAnimationComplete($item, newPosition);

});

} else {

$item.data('currentPosition', newPosition)

// Move the item to the 'hidden' position if hasn't been moved yet

// This is for the intitial setup

if ($item.data('oldPosition') === 0) {

$item.css({

'left': $item.data().left,

'width': $item.data().width,

'height': $item.data().height,

'top': $item.data().top,

'opacity': $item.data().opacity,

'z-index': $item.data().depth

});

}

}

}

/\*\*

\* This function is called once an item has finished animating to its

\* given position. Several different statements are executed here, such as

\* dealing with the animation queue

\*/

function itemAnimationComplete($item, newPosition) {

data.itemsAnimating--;

$item.data('currentPosition', newPosition);

// Keep track of what items came and left the center position,

// so we can fire callbacks when all the rotations are completed

if (newPosition === 0) {

data.currentCenterItem = $item;

}

// all items have finished their rotation, lets clean up

if (data.itemsAnimating === 0) {

data.carouselRotationsLeft -= 1;

data.currentlyMoving = false;

// If there are still rotations left in the queue, rotate the carousel again

// we pass in zero because we don't want to add any additional rotations

if (data.carouselRotationsLeft > 0) {

rotateCarousel(0);

// Otherwise there are no more rotations and...

} else {

// Reset the speed of the carousel to original

data.currentSpeed = options.speed;

data.currentCenterItem.addClass(options.activeClassName);

if (data.performingSetup === false) {

options.movedToCenter(data.currentCenterItem);

options.movedFromCenter(data.previousCenterItem);

}

data.performingSetup = false;

// reset & initate the autoPlay

autoPlay();

}

}

}

/\*\*

\* Function called to rotate the carousel the given number of rotations

\* in the given direciton. Will check to make sure the carousel should

\* be able to move, and then adjust speed and move items

\*/

function rotateCarousel(rotations) {

// Check to see that a rotation is allowed

if (data.currentlyMoving === false) {

// Remove active class from the center item while we rotate

data.currentCenterItem.removeClass(options.activeClassName);

data.currentlyMoving = true;

data.itemsAnimating = 0;

data.carouselRotationsLeft += rotations;

if (options.quickerForFurther === true) {

// Figure out how fast the carousel should rotate

if (rotations > 1) {

data.currentSpeed = options.speed / rotations;

}

// Assure the speed is above the minimum to avoid weird results

data.currentSpeed = (data.currentSpeed < 100) ? 100 : data.currentSpeed;

}

// Iterate thru each item and move it

for (var i = 0; i < data.totalItems; i++) {

var $item = $(data.items[i]);

var currentPosition = $item.data('currentPosition');

var newPosition;

if (data.currentDirection == 'forward') {

newPosition = currentPosition - 1;

} else {

newPosition = currentPosition + 1;

}

// We keep both sides as even as possible to allow circular rotation to work.

// We will "wrap" the item arround to the other side by negating its current position

var flankingAllowance = (newPosition > 0) ? data.rightItemsCount : data.leftItemsCount;

if (Math.abs(newPosition) > flankingAllowance) {

newPosition = currentPosition \* -1;

// If there's an uneven number of "flanking" items, we need to compenstate for that

// when we have an item switch sides. The right side will always have 1 more in that case

if (data.totalItems % 2 == 0) {

newPosition += 1;

}

}

moveItem($item, newPosition);

}

}

}

/\*\*

\* The event handler when an image within the carousel is clicked

\* This function will rotate the carousel the correct number of rotations

\* to get the clicked item to the center, or will fire the custom event

\* the user passed in if the center item is clicked

\*/

$(this).find('img').bind("click", function () {

var itemPosition = $(this).data().currentPosition;

if (options.imageNav == false) {

return;

}

// Don't allow hidden items to be clicked

if (Math.abs(itemPosition) >= options.flankingItems + 1) {

return;

}

// Do nothing if the carousel is already moving

if (data.currentlyMoving) {

return;

}

data.previousCenterItem = data.currentCenterItem;

// Remove autoplay

autoPlay(true);

options.autoPlay = 0;

var rotations = Math.abs(itemPosition);

if (itemPosition == 0) {

options.clickedCenter($(this));

} else {

// Fire the 'moving' callbacks

options.movingFromCenter(data.currentCenterItem);

options.movingToCenter($(this));

if (itemPosition < 0) {

data.currentDirection = 'backward';

rotateCarousel(rotations);

} else if (itemPosition > 0) {

data.currentDirection = 'forward';

rotateCarousel(rotations);

}

}

});

/\*\*

\* The user may choose to wrap the images is link tags. If they do this, we need to

\* make sure that they aren't active for certain situations

\*/

$(this).find('a').bind("click", function (event) {

var isCenter = $(this).find('img').data('currentPosition') == 0;

// should we disable the links?

if (options.linkHandling === 1 || // turn off all links

(options.linkHandling === 2 && !isCenter)) // turn off all links except center

{

event.preventDefault();

return false;

}

});

function nextItemFromCenter() {

var $next = data.currentCenterItem.next();

if ($next.length <= 0) {

$next = data.currentCenterItem.parent().children().first();

}

return $next;

}

function prevItemFromCenter() {

var $prev = data.currentCenterItem.prev();

if ($prev.length <= 0) {

$prev = data.currentCenterItem.parent().children().last();

}

return $prev;

}

/\*\*

\* Intiate a move of the carousel in either direction. Takes care of firing

\* the 'moving' callbacks

\*/

function moveOnce(direction) {

if (data.currentlyMoving === false) {

data.previousCenterItem = data.currentCenterItem;

options.movingFromCenter(data.currentCenterItem);

if (direction == 'backward') {

options.movingToCenter(prevItemFromCenter());

data.currentDirection = 'backward';

} else if (direction == 'forward') {

options.movingToCenter(nextItemFromCenter());

data.currentDirection = 'forward';

}

}

rotateCarousel(1);

}

/\*\*

\* Navigation with arrow keys

\*/

$(document).keydown(function (e) {

if (options.keyboardNav) {

// arrow left or up

if ((e.which === 37 && options.orientation == 'horizontal') || (e.which === 38 && options.orientation == 'vertical')) {

autoPlay(true);

options.autoPlay = 0;

moveOnce('backward');

// arrow right or down

} else if ((e.which === 39 && options.orientation == 'horizontal') || (e.which === 40 && options.orientation == 'vertical')) {

autoPlay(true);

options.autoPlay = 0;

moveOnce('forward');

}

// should we override the normal functionality for the arrow keys?

if (options.keyboardNavOverride && (

(options.orientation == 'horizontal' && (e.which === 37 || e.which === 39)) ||

(options.orientation == 'vertical' && (e.which === 38 || e.which === 40))

)) {

e.preventDefault();

return false;

}

}

});

/\*\*

\* Public API methods

\*/

this.reload = function (newOptions) {

if (typeof newOptions === "object") {

var combineDefaultWith = newOptions;

} else {

var combineDefaultWith = {};

}

options = $.extend({}, $.fn.waterwheelCarousel.defaults, newOptions);

initializeCarouselData();

data.itemsContainer.find('img').hide();

forceImageDimensionsIfEnabled();

preload(function () {

setOriginalItemDimensions();

preCalculatePositionProperties();

setupCarousel();

setupStarterRotation();

});

}

this.next = function () {

autoPlay(true);

options.autoPlay = 0;

moveOnce('forward');

}

this.prev = function () {

autoPlay(true);

options.autoPlay = 0;

moveOnce('backward');

}

this.reload(startingOptions);

return this;

};

$.fn.waterwheelCarousel.defaults = {

// number tweeks to change apperance

startingItem: 1, // item to place in the center of the carousel. Set to 0 for auto

separation: 175, // distance between items in carousel

separationMultiplier: 0.6, // multipled by separation distance to increase/decrease distance for each additional item

horizonOffset: 0, // offset each item from the "horizon" by this amount (causes arching)

horizonOffsetMultiplier: 1, // multipled by horizon offset to increase/decrease offset for each additional item

sizeMultiplier: 0.7, // determines how drastically the size of each item changes

opacityMultiplier: 0.8, // determines how drastically the opacity of each item changes

horizon: 0, // how "far in" the horizontal/vertical horizon should be set from the container wall. 0 for auto

flankingItems: 3, // the number of items visible on either side of the center

// animation

speed: 300, // speed in milliseconds it will take to rotate from one to the next

animationEasing: 'linear', // the easing effect to use when animating

quickerForFurther: true, // set to true to make animations faster when clicking an item that is far away from the center

edgeFadeEnabled: false, // when true, items fade off into nothingness when reaching the edge. false to have them move behind the center image

// misc

linkHandling: 2, // 1 to disable all (used for facebox), 2 to disable all but center (to link images out)

autoPlay: 0, // indicate the speed in milliseconds to wait before autorotating. 0 to turn off. Can be negative

orientation: 'horizontal', // indicate if the carousel should be 'horizontal' or 'vertical'

activeClassName: 'carousel-center', // the name of the class given to the current item in the center

keyboardNav: false, // set to true to move the carousel with the arrow keys

keyboardNavOverride: true, // set to true to override the normal functionality of the arrow keys (prevents scrolling)

imageNav: true, // clicking a non-center image will rotate that image to the center

// preloader

preloadImages: true, // disable/enable the image preloader.

forcedImageWidth: 0, // specify width of all images; otherwise the carousel tries to calculate it

forcedImageHeight: 0, // specify height of all images; otherwise the carousel tries to calculate it

// callback functions

movingToCenter: $.noop, // fired when an item is about to move to the center position

movedToCenter: $.noop, // fired when an item has finished moving to the center

clickedCenter: $.noop, // fired when the center item has been clicked

movingFromCenter: $.noop, // fired when an item is about to leave the center position

movedFromCenter: $.noop // fired when an item has finished moving from the center

};

})(jQuery);

$(document).ready(function () {

var carousel = $('.carousel').waterwheelCarousel();

});

function shuffle(array) {

var i = 0, j = 0, temp = null; //shuffle the sorting array

for (i = array.length - 1; i > 0; i -= 1) {

j = Math.floor(Math.random() \* (i + 1))

temp = array[i];

array[i] = array[j];

array[j] = temp;

temp = array[i].x;

array[i].x = array[j].x;

array[j].x = temp;

}

return array;

}

function randVal(min, max) {

return (Math.floor(Math.random() \* (max - min + 1) + min)); //returns the max and min to get ready for sorting

}

function randFloat(min, max) {

return (Math.random() \* (max - min + 1) + min);

}

function start() {

myCanvas = document.getElementById("myCanvas"); //starts sorting

ctx = myCanvas.getContext("2d");

var widthRatio = 0.9;

var heightRatio = 0.50;

myCanvas.width = window.innerWidth \* widthRatio;

myCanvas.height = window.innerHeight \* heightRatio;

var barSettings = {

width: 25,

heightMultiplier: 20,

margin: 7

}

var delay = {

checkSortDelay: 1000,

swapAnimationDelay: 33,

swapTimeoutDelay: 1500

}

document.getElementById("animSpeed").onchange = function () {

delay.swapAnimationDelay = Math.abs(document.getElementById("animSpeed").value); delay.swapTimeoutDelay = (barSettings.width + barSettings.margin) \* delay.swapAnimationDelay + 700;

//alert(delay.swapTimeoutDelay);

}

var swapInterval = null;

var swappingFlag = false;

var colorsArray = ["yellow", "black", "red", "green", "blue", "orange", "purple", "lightgreen", "pink"]

var barsArray = [];

document.getElementById("shuffleArray").onclick = function () {

barsArray = shuffle(barsArray);

drawBars();

}

function descending(a, b) { return a < b; }

function ascending(a, b) { return a > b; }

document.getElementById("bubbleSort").onclick = function () {

document.getElementById("shuffleArray").disabled = true;

document.getElementById("bubbleSort").disabled = true;

document.getElementById("selectOrder").disabled = true;

func = document.getElementById("selectOrder").value;

bubbleSort(barsArray, eval(func));

sortInterval = setInterval(function () {

sorted = true;

for (i = 0; i < barsArray.length - 1; i++) {

if (barsArray[i].x > barsArray[i + 1].x) {

sorted = false;

}

}

if (sorted) {

clearInterval(sortInterval);

document.getElementById("shuffleArray").disabled = false; document.getElementById("bubbleSort").disabled = false; document.getElementById("selectOrder").disabled = false; document.getElementById("info").innerHTML = "All Done!";

}

}, delay.checkSortDelay);

}

xBuffer = barSettings.margin \* 2;

for (i = 1; i < 10; i++) {

barsArray.push({

value: i,

width: barSettings.width,

height: barSettings.heightMultiplier \* i,

x: xBuffer,

y: myCanvas.height - barSettings.heightMultiplier \* i - 10,

color: colorsArray[i % colorsArray.length]

});

xBuffer += barSettings.width + barSettings.margin;

}

function drawBars() {

ctx.fillStyle = "lightblue"; ctx.fillRect(0, 0, myCanvas.width, myCanvas.height);

xBuffer = barSettings.margin \* 2;

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

bar = barsArray[i];

ctx.beginPath();

ctx.fillStyle = bar.color;

ctx.rect(bar.x, bar.y, bar.width, bar.height);

ctx.fill();

ctx.font = "20px Times New Roman";

ctx.fillStyle = "black"; ctx.fillText(bar.value, bar.x + barSettings.width / 4, bar.y - barSettings.margin);

ctx.closePath();

}

}

function swapBars(barA, barB, compFunc) {

function swapAnimation() { //animation that swaps bars that is sorting

if ((barA.x >= xB || barB.x <= xA) || (xFakeA >= xB || xFakeB <= xA)) {

clearInterval(swapInterval);

swapInterval = null;

swappingFlag = false;

barA.color = cA;

barB.color = cB;

}

else {

if (compFunc(barA.value, barB.value)) {

barA.x++;

barB.x--;

}

xFakeA++;

xFakeB--;

}

drawBars();

}

if (!swapInterval) {

xA = barA.x;

xB = barB.x;

xFakeA = barA.x;

xFakeB = barB.x;

cA = barA.color;

cB = barB.color;

swapColor = (compFunc(barA.value, barB.value) ? "white" : "gray");

barA.color = swapColor;

barB.color = swapColor;

swappingFlag = true;

document.getElementById("info").innerHTML = barA.value + ((document.getElementById("selectOrder").value === "ascending") ? " > " : " < ") + barB.value + ((compFunc(barA.value, barB.value)) ? " --> SWAP!" : " --> NO SWAP!");

swapInterval = setInterval(swapAnimation, delay.swapAnimationDelay);

}

else {

setTimeout(swapBars.bind(null, barA, barB, compFunc), delay.swapTimeoutDelay);

}

//swapInterval=setInterval(swapAnimation.bind(null,barsArray[0],barsArray[1]),33);

}

function bubbleSort(a, compFunc) {

var swapped;

//do

for (j = 0; j < a.length - 1; j++) {

swapped = false;

for (var i = 0; i < a.length - 1 - j; i++) {

// run one less iteration each round as the right side is already sorted

//document.getElementById("debug").innerHTML+=a[i].value+" > "+a[i+1].value+" "+((a[i].value > a[i+1].value)? "V":"X")+";\n";

//if (a[i].value > a[i+1].value)

swapBars(a[i], a[i + 1], compFunc);

if (compFunc(a[i].value, a[i + 1].value)) {

var temp = a[i];

a[i] = a[i + 1];

a[i + 1] = temp;

swapped = true;

}

}

if (!swapped) {

// no swaps were made in the inner loop --> all sorted

break;

}

}// while (swapped);

}

drawBars();

}

window.onload = start;

function login2()/\*function to check userid & password\*/

{

$.get("XmlLogin.xml", MyCallback);

}

function MyCallback(data , status) { //part of login

var parser = new DOMParser();

xmlDoc = parser.parseFromString(data, "text/xml");

// var Users = xmlDoc.getElementsByTagName("userid")[0].childNodes[0].nodeValue;

var sName = xmlDoc.getElementById("userID").value;

var sPassword = XmlDoc.getElementById("password").value;

for (var i = 0; i <= 5; i++) {

var name = xmlDoc[i].getElementsByTagName("userID")[0].firstChild.data;

var password = xmlDoc[i].getElementsByTagName("password")[0].firstChild.data;

if (sName === name) {

UsernameFound = true; //checks name field in xml

}

if (sPassword === password) { //checks password field in xml

PasswordFound = true;

}

if ((sName === name) && (sPassword === password)) { //validate if user and password is correct

var UserFound = true;

ClassNumber = i;

}

else {

i += 1;

}

if (UsernameFound && PasswordFound === true) { //when username and password is correct output

window.open('HomePage.html');

break;

}

else {

if (UsernameFound === false) {

alert('Wrong username or password!');

break;

}

if (PasswordFound === false) {

alert('Wrong username or password!');

break;

}

}

}

}

function shuffle(array) {

var i = 0, j = 0, temp = null;

for (i = array.length - 1; i > 0; i -= 1) {

j = Math.floor(Math.random() \* (i + 1))

temp = array[i];

array[i] = array[j];

array[j] = temp;

temp = array[i].x;

array[i].x = array[j].x;

array[j].x = temp;

}

return array;

}

function randVal(min, max) {

return (Math.floor(Math.random() \* (max - min + 1) + min));

}

function randFloat(min, max) {

return (Math.random() \* (max - min + 1) + min);

}

function start() {

myCanvas = document.getElementById("myCanvas");

ctx = myCanvas.getContext("2d");

var widthRatio = 0.9;

var heightRatio = 0.50;

myCanvas.width = window.innerWidth \* widthRatio;

myCanvas.height = window.innerHeight \* heightRatio;

var barSettings = {

width: 25,

heightMultiplier: 20,

margin: 7

}

var delay = {

checkSortDelay: 1000,

swapAnimationDelay: 33,

swapTimeoutDelay: 1500

}

document.getElementById("animSpeed").onchange = function () {

delay.swapAnimationDelay = Math.abs(document.getElementById("animSpeed").value); delay.swapTimeoutDelay = (barSettings.width + barSettings.margin) \* delay.swapAnimationDelay + 700;

//alert(delay.swapTimeoutDelay);

}

var swapInterval = null;

var swappingFlag = false;

var colorsArray = ["yellow", "black", "red", "green", "blue", "orange", "purple", "lightgreen", "pink"]

var barsArray = [];

document.getElementById("shuffleArray").onclick = function () {

barsArray = shuffle(barsArray);

drawBars();

}

function descending(a, b) { return a < b; }

function ascending(a, b) { return a > b; }

document.getElementById("bubbleSort").onclick = function () {

document.getElementById("shuffleArray").disabled = true;

document.getElementById("bubbleSort").disabled = true;

document.getElementById("selectOrder").disabled = true;

func = document.getElementById("selectOrder").value;

bubbleSort(barsArray, eval(func));

sortInterval = setInterval(function () {

sorted = true;

for (i = 0; i < barsArray.length - 1; i++) {

if (barsArray[i].x > barsArray[i + 1].x) {

sorted = false;

}

}

if (sorted) {

clearInterval(sortInterval);

document.getElementById("shuffleArray").disabled = false; document.getElementById("bubbleSort").disabled = false; document.getElementById("selectOrder").disabled = false; document.getElementById("info").innerHTML = "All Done!";

}

}, delay.checkSortDelay);

}

xBuffer = barSettings.margin \* 2;

for (i = 1; i < 10; i++) {

barsArray.push({

value: i,

width: barSettings.width,

height: barSettings.heightMultiplier \* i,

x: xBuffer,

y: myCanvas.height - barSettings.heightMultiplier \* i - 10,

color: colorsArray[i % colorsArray.length]

});

xBuffer += barSettings.width + barSettings.margin;

}

function drawBars() {

ctx.fillStyle = "lightblue"; ctx.fillRect(0, 0, myCanvas.width, myCanvas.height);

xBuffer = barSettings.margin \* 2;

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

bar = barsArray[i];

ctx.beginPath();

ctx.fillStyle = bar.color;

ctx.rect(bar.x, bar.y, bar.width, bar.height);

ctx.fill();

ctx.font = "20px Times New Roman";

ctx.fillStyle = "black"; ctx.fillText(bar.value, bar.x + barSettings.width / 4, bar.y - barSettings.margin);

ctx.closePath();

}

}

function bubbleSort(a) {

var swapped;

do {

swapped = false;

for (var i = 0; i < a.length - 1; i++) {

if (a[i] > a[i + 1]) {

var temp = a[i];

a[i] = a[i + 1];

a[i + 1] = temp;

swapped = true;

}

}

} while (swapped);

}

function swapBars(barA, barB, compFunc) {

function swapAnimation() {

if ((barA.x >= xB || barB.x <= xA) || (xFakeA >= xB || xFakeB <= xA)) {

clearInterval(swapInterval);

swapInterval = null;

swappingFlag = false;

barA.color = cA;

barB.color = cB;

}

else {

if (compFunc(barA.value, barB.value)) {

barA.x++;

barB.x--;

}

xFakeA++;

xFakeB--;

}

drawBars();

}

if (!swapInterval) {

xA = barA.x;

xB = barB.x;

xFakeA = barA.x;

xFakeB = barB.x;

cA = barA.color;

cB = barB.color;

swapColor = (compFunc(barA.value, barB.value) ? "white" : "gray");

barA.color = swapColor;

barB.color = swapColor;

swappingFlag = true;

document.getElementById("info").innerHTML = barA.value + ((document.getElementById("selectOrder").value === "ascending") ? " > " : " < ") + barB.value + ((compFunc(barA.value, barB.value)) ? " --> SWAP!" : " --> NO SWAP!");

swapInterval = setInterval(swapAnimation, delay.swapAnimationDelay);

}

else {

setTimeout(swapBars.bind(null, barA, barB, compFunc), delay.swapTimeoutDelay);

}

//swapInterval=setInterval(swapAnimation.bind(null,barsArray[0],barsArray[1]),33);

}

/\* function bubbleSort(a, compFunc) {

var swapped;

//do

for (j = 0; j < a.length - 1; j++) {

swapped = false;

for (var i = 0; i < a.length - 1 - j; i++) {

// run one less iteration each round as the right side is already sorted

//document.getElementById("debug").innerHTML+=a[i].value+" > "+a[i+1].value+" "+((a[i].value > a[i+1].value)? "V":"X")+";\n";

//if (a[i].value > a[i+1].value)

swapBars(a[i], a[i + 1], compFunc);

if (compFunc(a[i].value, a[i + 1].value)) {

var temp = a[i];

a[i] = a[i + 1];

a[i + 1] = temp;

swapped = true;

}

}

if (!swapped) {

// no swaps were made in the inner loop --> all sorted

break;

}

}// while (swapped);

}\*/

drawBars();

}

window.onload = start;