**LOL 2.0**

**code**

// ======= ======= ======= ======= ======= script.js ======= ======= ======= ======= =======

// ======= ======= ======= ======= ======= script.js ======= ======= ======= ======= =======

// ======= ======= ======= ======= ======= script.js ======= ======= ======= ======= =======

var displayItems = {

studio: { itemName: "studio", itemText: "Studio View", can:null, ctx:null, canL:10, canT:280, canW:720, canH:405 },

monitor: { itemName: "monitor", itemText: "Monitor", can:null, ctx:null, canL:740, canT:10, canW:384, canH:216 },

shop: { itemName: "shop", itemText: "Shop Menu" },

lessons: { itemName: "lessons", itemText: "Lesson Menu" },

gridTop: 0,

gridLeft: 0,

warningFlag: false

}

var gearItems = {

diffusion: { itemName:"diffusion", itemText:"diffusion", itemType:"gel", image:"diff1\_0.png" },

Fresnel150W: { itemName:"Fresnel 150W", itemText:"Fresnel150W", itemType:"light", image:"f150\_0.png" },

Fresnel300W: { itemName:"Fresnel 300W", itemText:"Fresnel300W", itemType:"light", image:"f300\_0.png" },

Fresnel650W: { itemName:"Fresnel 650W", itemText:"Fresnel650W", itemType:"light", image:"f650\_0.png" },

Fresnel1000W: { itemName:"Fresnel 1000W", itemText:"Fresnel1000W", itemType:"light", image:"f1000\_0.png" },

Floppy: { itemName:"Floppy", itemText:"Floppy", itemType:"controller", image:"floppy\_0.png" },

LEDLight: { itemName:"LED Light", itemText:"LEDLight", itemType:"light", image:"LED\_0.png" },

HalfStopScrim: { itemName:"Half Stop Scrim", itemText:"HalfStopScrim", itemType:"scrim", image:"scrim1\_0.png" },

FullStopScrim: { itemName:"Full Stop Scrim", itemText:"FullStopScrim", itemType:"scrim", image:"scrim2\_0.png" },

GraduatedScrim: { itemName:"Graduated Scrim", itemText:"GraduatedScrim", itemType:"scrim", image:"scrim3\_0.png" },

Softlight: { itemName:"Softlight", itemText:"Softlight", itemType:"light", image:"softlight\_0.png" }

}

// ======= ======= ======= INITIALIZE ======= ======= =======

// ======= ======= ======= INITIALIZE ======= ======= =======

// ======= ======= ======= INITIALIZE ======= ======= =======

var clientApp = {

pages: null,

lessons: null,

activePage: null,

activeActor: null,

activeLesson: null,

gearItems: gearItems,

displayItems: displayItems,

monitorImages: [],

studioImages: [],

// ======= initialize =======

initialize: function() {

console.log("initialize");

this.targets = initTargets();

this.items = initItems(this.targets);

this.pages = initPages(this.items, this.targets);

this.lessons = initLessons();

this.activePage = this.pages.page\_0\_0;

this.activeLesson = this.lessons.lesson\_0;

this.activeActor = null;

this.makeLessonMenu(3);

this.makeGearMenu();

this.activatePrevNext();

this.activateMenuItems("lessonMenu");

this.activateDisplayItems();

},

// ======= makeLessonMenu =======

makeLessonMenu: function(pendingLesson) {

console.log("\n\*\*\*\*\*\*\* makeLessonMenu");

var index = -1;

var menuHtml = "<ul id='lessonMenu'>";

$.each(this.lessons, function(key, lesson) {

index++;

// == hide lessons not yet built (development)

if (index > pendingLesson) {

var menuState = "lessonItem inactive";

} else {

var menuState = "lessonItem active";

}

menuHtml += "<li><div id='" + key + "' class='" + menuState + "'>";

menuHtml += "<span class='menu\_title\_active'>" + lesson.lessonIndex + " - " +

lesson.lessonTitle + "</span>"

menuHtml += "<span class='menu\_text\_active'>" + lesson.lessonSubtitle + "</span>"

menuHtml += "</div>";

});

menuHtml += "</ul>";

$('#lessonMenuDisplay').html(menuHtml);

},

// ======= makeGearMenu =======

makeGearMenu: function() {

console.log("makeGearMenu");

var index = -1;

var menuHtml = "<ul id='gearMenu'>";

$.each(this.gearItems, function(key, item) {

index++;

menuHtml += "<li><div id='" + key + "\_" + index + "' class='gearItem'>";

menuHtml += "<div class='gearImage light' style='background-image:url(images/" + item.image + ");background-color:#ccc;background-repeat:no-repeat;background-size:100%;'></div>";

menuHtml += "<div><span class='gear\_label\_active'>" + item.itemName + "</span>";

menuHtml += "<span class='gear\_text\_active'>" + item.itemType + "</span></div>";

menuHtml += "</li>";

});

menuHtml += "</ul>";

$('#storeroomMenuDisplay').html(menuHtml);

},

// ======= ======= ======= USER ACTION ======= ======= =======

// ======= ======= ======= USER ACTION ======= ======= =======

// ======= ======= ======= USER ACTION ======= ======= =======

// ======= makeLessonPage =======

makeLessonPage: function(lessonEl) {

console.log("\n \*\*\*\*\*\*\* makeLessonPage \*\*\*\*\*\*\*");

this.initLessonCanvases();

this.clearPageElements();

this.makeLessonCanvases();

this.makeLessonItems();

this.activateLessonItems();

this.makeLessonText(lessonEl);

},

// ======= initLessonCanvases =======

initLessonCanvases: function(lesson) {

console.log("initLessonCanvases");

// == get device pixel ratio

var devicePixelRatio = window.devicePixelRatio || 1;

if (!this.displayItems["studio"].can) {

// == make canvas and context objects

var canvases = ["studio", "monitor"];

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

var can = document.getElementById(canvases[i] + "Canvas");

var ctx = can.getContext('2d');

// == calculate ratio for normal/retina displays

var backingStoreRatio = ctx.webkitBackingStorePixelRatio ||

ctx.mozBackingStorePixelRatio ||

ctx.msBackingStorePixelRatio ||

ctx.oBackingStorePixelRatio ||

ctx.backingStorePixelRatio || 1;

var ratio = devicePixelRatio / backingStoreRatio;

var width = can.offsetWidth;

var height = can.offsetHeight;

can.width = width \* ratio;

can.height = height \* ratio;

can.style.width = width + "px";

can.style.height = height + "px";

can.getContext('2d').scale(2,2)

// == store can/ctx on app object

this.displayItems[canvases[i]].can = can;

this.displayItems[canvases[i]].ctx = ctx;

}

}

},

// ======= makeLessonCanvases =======

makeLessonCanvases: function(lesson) {

console.log("makeLessonCanvases");

clientApp.studioImages = [];

clientApp.monitorImages = [];

var page = clientApp.activePage;

if (page.studio.image) {

$('#studioCanvas, #studio\_tab').fadeIn('fast', function() {

// console.log("studio IN");

})

$('#hover\_text').addClass('loading');

$('#hover\_text').text("Images loading... please wait");

// == allows "sticky" message for image loading (won't clear on mouseleave)

clientApp.displayItems.warningFlag = true;

loadCanvasImages("studio");

} else {

$('#studioCanvas, #studio\_tab').fadeOut('fast', function() {

// console.log("studio OUT");

})

}

if (page.monitor.image) {

$('#monitorCanvas, #monitor\_box').fadeIn('fast', function() {

// console.log("monitor IN");

})

loadCanvasImages("monitor");

} else {

$('#monitorCanvas, #monitor\_box').fadeOut('fast', function() {

// console.log("monitor OUT");

})

}

// ======= loadCanvasImages =======

function loadCanvasImages(canvas) {

console.log("loadCanvasImages");

// == get frameset and canvas params

var folder = page[canvas].folder;

var imageName = page[canvas].image;

var initFrame = page[canvas].initFrame;

var can = clientApp.displayItems[canvas].can;

var ctx = clientApp.displayItems[canvas].ctx;

var canW = clientApp.displayItems[canvas].canW;

var canH = clientApp.displayItems[canvas].canH;

// == set start/end frame indexes

if (page.studio.matrix) {

var matrix = page.studio.matrix;

var YfilesArray = [];

var startFrameX = page[canvas].startFrame;

var endFrameX = page[canvas].endFrame + 1;

var startFrameY = page[canvas].startFrame;

var endFrameY = page[canvas].endFrame + 1;

} else {

var matrix = null;

var startFrameX = page[canvas].startFrame;

var endFrameX = page[canvas].endFrame + 1;

}

// == start image load self-invoking iterative function

loadNextImage(0, 0);

// ======= loadNextImage =======

function loadNextImage(Xindex, Yindex) {

console.log("loadNextImage");

// == make image elements; assure image loading via timeout

if (Xindex < endFrameX) {

if (matrix) {

if (Yindex < endFrameY) {

var canvasImage = new Image();

canvasImage.id = imageName + "\_" + Xindex + Yindex;

canvasImage.src = "images/"+folder+"/"+imageName+"\_"+Xindex+Yindex+".png";

canvasImage.onload = function() {

setTimeout(function(){

YfilesArray.push(canvasImage);

Yindex++;

loadNextImage(Xindex, Yindex);

}, 10);

}

} else {

if (canvas == "studio") {

clientApp.studioImages.push(YfilesArray);

} else {

clientApp.monitorImages.push(YfilesArray);

}

Xindex++;

Yindex = 0;

YfilesArray = [];

loadNextImage(Xindex, Yindex);

}

} else {

var canvasImage = new Image();

canvasImage.id = imageName + "\_" + Xindex;

canvasImage.src = "images/" + imageName + "\_" + Xindex + ".png";

canvasImage.onload = function() {

setTimeout(function(){

if (canvas == "studio") {

clientApp.studioImages.push(canvasImage);

} else {

clientApp.monitorImages.push(canvasImage);

}

Xindex++;

loadNextImage(Xindex, Yindex);

}, 10);

}

}

// == display init image (usually first in frameset)

} else {

// == clear load warning (enables mouseleave message clearing)

$('#hover\_text').fadeOut("fast", function() {

clientApp.displayItems.warningFlag = false;

$('#hover\_text').removeClass('loading');

$('#hover\_text').text('');

$('#hover\_text').css('display', 'block');

});

if (matrix) {

if (canvas == "studio") {

var initImage = clientApp.studioImages[initFrame][initFrame];

} else {

if (page.monitor.image) {

var initImage = clientApp.monitorImages[initFrame][initFrame];

}

}

} else {

if (canvas == "studio") {

var initImage = clientApp.studioImages[initFrame];

} else {

var initImage = clientApp.monitorImages[initFrame];

}

}

ctx.clearRect(0, 0, canW, canH);

// == drawImage(img, imgX, imgY, imgW, imH, canX, canY, canW, canH);

if (initImage) {

ctx.drawImage(initImage, 0, 0, 720, 405, 0, 0, canW, canH);

ctx.save();

}

}

}

}

},

// ======= makeLessonItems =======

makeLessonItems: function() {

console.log("makeLessonItems");

var page = this.activePage;

var setups = page.SetupItem;

var actors = page.ActorItems;

var groups = page.GroupItems;

var guides = page.guides;

var targets = page.pageTargets;

var gridders = page.GridItems;

var lessonItemsArray = [groups, gridders, actors, targets];

// ======= make html for lesson items =======

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

if ((lessonItemsArray[i]) && (lessonItemsArray[i].length > 0)) {

makeItemEls(lessonItemsArray[i]);

if (guides.length > 0) {

this.makeItemGuides(lessonItemsArray[i]);

}

}

}

// ======= make html for setup items =======

if (setups.item) {

makeItemEls([setups.item]); // makeItemEls expects an array

}

// ======= makeItemEls =======

function makeItemEls(items) {

console.log("makeItemEls");

var item, itemType, urlString, newDiv, target, gridStartL, border;

clientApp.displayItems.gridLeft = clientApp.displayItems.studio.canL + 10;

clientApp.displayItems.gridTop = clientApp.displayItems.studio.canT + 10;

$('#grid').css('left', clientApp.displayItems.gridLeft + 'px');

$('#grid').css('top', clientApp.displayItems.gridTop + 'px');

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

item = items[i];

itemType = item.itemType;

switch(itemType) {

case "target":

newDiv = makeItemHtml(item);

item.itemEl = newDiv;

locatePageTarget(item, newDiv);

break;

case "gridItem":

newDiv = makeItemHtml(item);

item.itemEl = newDiv;

locateGridItem(item, newDiv, i);

break;

case "actor":

newDiv = makeItemHtml(item);

item.itemEl = newDiv;

locateNewActor(item, newDiv);

break;

case "setup":

newDiv = makeItemHtml(item);

item.itemEl = newDiv;

locateNewSetup(item, newDiv);

// == make html for setup targets and controls (part of setup item)

if (setups.targets) {

for (var j = 0; j < setups.targets.length; j++) {

target = setups.targets[j];

newDiv = makeSetupPartHtml(target);

target.itemEl = newDiv;

locateSetupParts(target, item, newDiv);

}

}

console.log("setups.controls:", setups.controls);

if (setups.controls) {

for (var j = 0; j < setups.controls.length; j++) {

control = setups.controls[j];

newDiv = makeSetupPartHtml(control);

control.itemEl = newDiv;

locateSetupParts(control, item, newDiv);

}

}

break;

}

}

// ======= locateNewActor =======

function locateNewActor(item, newDiv) {

console.log("locateNewActor");

newDiv.style.left = item.initLTWH.L + displayItems.studio.canL + 'px';

newDiv.style.top = item.initLTWH.T + displayItems.studio.canT + 'px';

newDiv.style.width = item.initLTWH.W + 'px';

newDiv.style.height = item.initLTWH.H + 'px';

newDiv.style.zIndex = 4;

$('#actors').append(newDiv);

}

// ======= locateNewSetup =======

function locateNewSetup(item, newDiv) {

console.log("locateNewSetup");

newDiv.style.left = item.initLTWH.L + displayItems.studio.canL + 'px';

newDiv.style.top = item.initLTWH.T + displayItems.studio.canT + 'px';

newDiv.style.width = item.initLTWH.W + 'px';

newDiv.style.height = item.initLTWH.H + 'px';

newDiv.style.zIndex = 2;

$('#setup').append(newDiv);

}

// ======= locateSetupParts =======

function locateSetupParts(setupPart, item, newDiv) {

console.log("locateSetupParts");

// == setup target and control initLTWH is offset from setup item

newDiv.style.left = item.initLTWH.L+displayItems.studio.canL+setupPart.initLTWH.L+'px';

newDiv.style.top = item.initLTWH.T+displayItems.studio.canT+setupPart.initLTWH.T+'px';

if (item.itemImage.image) {

newDiv.style.width = item.itemImage.image.naturalWidth + 'px';

newDiv.style.height = item.itemImage.image.naturalHeight + 'px';

} else {

newDiv.style.width = setupPart.initLTWH.W + 'px';

newDiv.style.height = setupPart.initLTWH.H + 'px';

}

newDiv.style.zIndex = 4;

$('body').append(newDiv);

}

// ======= locateGridItem =======

function locateGridItem(item, newDiv, itemIndex) {

console.log("locateGridItem");

// == check vertical space for new item; move to right if not

if (itemIndex == 0) {

var gridL = clientApp.displayItems.gridLeft;

var gridT = clientApp.displayItems.gridTop;

} else {

var gridL = clientApp.displayItems.gridLeft;

var gridT = clientApp.displayItems.gridTop + item.initLTWH.H + 10;

clientApp.displayItems.gridTop = gridT;

if ((gridT + item.initLTWH.H + 10) > (clientApp.displayItems.studio.canT +

clientApp.displayItems.studio.canH)) {

var gridL = clientApp.displayItems.gridLeft + item.initLTWH.W + 10;

var gridT = clientApp.displayItems.studio.canT + 10;

clientApp.displayItems.gridLeft = gridL;

clientApp.displayItems.gridTop = gridT;

}

}

// == locate item on new grid values

newDiv.style.left = gridL + 'px';

newDiv.style.top = gridT + 'px';

newDiv.style.width = item.initLTWH.W + 'px';

newDiv.style.height = item.initLTWH.H + 'px';

newDiv.style.zIndex = 4;

item.initLTWH.L = gridL;

item.initLTWH.T = gridT;

$('#grid').append(newDiv);

}

// ======= locatePageTarget =======

function locatePageTarget(target, newDiv) {

console.log("locatePageTarget");

// == pageTarget initLTWH is absolute (page)

newDiv.style.left = displayItems.studio.canL + target.initLTWH.L + 'px';

newDiv.style.top = displayItems.studio.canT + target.initLTWH.T + 'px';

newDiv.style.width = target.initLTWH.W + 'px';

newDiv.style.height = target.initLTWH.H + 'px';

newDiv.style.zIndex = 3;

$('body').append(newDiv);

}

// ======= locateControlOutline =======

function locateControlOutline(control, item, newDiv) {

console.log("locateControlOutline");

// == setupTarget initLTWH is offset from setup item

newDiv.style.left = item.initLTWH.L+displayItems.studio.canL+setupPart.initLTWH.L+'px';

newDiv.style.top = item.initLTWH.T+displayItems.studio.canT+setupPart.initLTWH.T+'px';

if (item.itemImage.image) {

newDiv.style.width = item.itemImage.image.naturalWidth + 'px';

newDiv.style.height = item.itemImage.image.naturalHeight + 'px';

} else {

newDiv.style.width = setupPart.initLTWH.W + 'px';

newDiv.style.height = setupPart.initLTWH.H + 'px';

}

newDiv.style.zIndex = 4;

$('body').append(newDiv);

}

// ======= makeItemHtml =======

function makeItemHtml(item) {

console.log("makeItemHtml");

newDiv = document.createElement('div');

newDiv.id = item.itemId;

newDiv.classList.add(item.itemType);

newDiv.style.position = "absolute";

if (item.itemImage) {

urlString = "url('images/" + item.itemImage + ".png') 0 0";

newDiv.style.background = urlString;

newDiv.style.backgroundSize = item.initLTWH.W + 'px ' + item.initLTWH.H + 'px';

}

return newDiv;

}

// ======= makeSetupPartHtml =======

function makeSetupPartHtml(item) {

console.log("makeSetupPartHtml");

newDiv = document.createElement('div');

newDiv.id = item.itemId;

newDiv.classList.add(item.itemType);

newDiv.style.position = "absolute";

if (item.itemImage.image) {

urlString = "url('images/" + item.itemImage.image + "\_" + item.itemImage.startFrame + ".png') 0 0";

newDiv.style.background = urlString;

newDiv.style.backgroundRepeat = "no-repeat";

newDiv.style.backgroundSize = item.itemImage.image.naturalWidth + 'px ' + item.itemImage.image.naturalHeight + 'px';

}

return newDiv;

}

// ======= makeControlOutline =======

function makeControlOutline(control, item) {

console.log("makeControlOutline");

newDiv = document.createElement('div');

newDiv.id = control.itemId + "\_border";

newDiv.classList.add("control-border");

newDiv.style.position = "absolute";

return newDiv;

}

}

},

// ======= makeItemGuides =======

makeItemGuides: function(items) {

console.log("makeItemGuides");

var item, itemId;

var guides = this.activePage.guides;

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

item = items[i];

itemId = items[i].itemId;

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

var guideId = guides[i].itemId;

if (guideId == itemId) {

makeItemGuide(item);

}

}

// ======= makeItemGuide =======

function makeItemGuide(item) {

console.log("makeItemGuide");

// == make svg guide elements

var guidesEl = document.getElementById("guides");

guidesEl.style.position = "absolute";

guidesEl.style.left = (item.bounds.L + displayItems.studio.canL + 5) + 'px';

guidesEl.style.top = (item.bounds.T + displayItems.studio.canT + 25) + 'px';

guidesEl.style.width = item.bounds.W + 'px';

guidesEl.style.height = item.bounds.H + 'px';

guidesEl.style.zIndex = 1;

var data = [[0, item.bounds.H], [item.bounds.W, 0]];

var line = d3.line(data);

var lineGenerator = d3.line();

var pathString = lineGenerator(data);

// == make svg line element

var svgEl = d3.select(guidesEl)

.append("svg")

.attr("width", item.bounds.W)

.attr("height", item.bounds.H);

svgEl.append("path");

d3.select('path')

.attr('d', pathString)

.style("stroke", "red")

.style("stroke-weight", "2")

.style("fill", "none");

}

}

},

// ======= makeLessonText =======

makeLessonText: function(lessonEl) {

console.log("makeLessonText");

var lessonSelectedId = $(lessonEl).attr('id') + "\_selected";

var titleText = clientApp.activeLesson.lessonIndex + " - " + clientApp.activeLesson.lessonTitle;

var subTitleText = clientApp.activeLesson.lessonSubtitle;

$('#selectedLesson').children('span').eq(0).text(titleText);

$('#selectedLesson').children('span').eq(1).text(subTitleText);

$("#lessonTextDisplay").fadeIn("fast", function() {

// console.log("MENU opacity 0");

});

$("#lessonMenuDisplay").fadeOut("fast", function() {

// console.log("MENU opacity 0");

});

$("#storeroomMenuDisplay").fadeOut("fast", function() {

// console.log("STOREROOM opacity 0");

});

// == update lesson text and activate prev/next

this.updateLessonText();

},

// ======= updateLessonText =======

updateLessonText: function(errorText) {

console.log("updateLessonText");

// == no page found... display msg

if (errorText) {

var lessonText = errorText;

} else {

var lessonText = clientApp.activePage.pageText;

}

// == remove previous lesson text

if ($('#lessonText').children('p').html()) {

$('#lessonText').animate({

height: 0,

opacity: 0

}, 500, function() {

console.log("collapsed");

// == replace prev lesson text with new text

$('#lessonText').children('p').html(lessonText);

$('#lessonText').animate({

height: "200px",

opacity: 1.0

}, 500, function() {

console.log("expanded");

});

});

} else {

// == replace lessons list with lesson text

$('#lessonText').children('p').html(lessonText);

$('#lessonText').removeClass('hide');

$('#lessonText').css('display, block');

$('#lessonText').animate({

height: "200px",

opacity: 1.0

}, 500, function() {

console.log("expanded");

});

}

},

// ======= selectSectionItem =======

selectSectionItem: function(item) {

console.log("selectSectionItem");

var itemId = $(item).attr('id');

switch (itemId) {

case "lessons":

deselectItem("shop");

selectItem(itemId);

$( "#storeroomMenuDisplay" ).fadeOut( "fast", function() {

// console.log("STOREROOM opacity 0");

});

$( "#lessonTextDisplay" ).fadeOut( "fast", function() {

// console.log("LESSON opacity 0");

});

$( "#lessonMenuDisplay" ).fadeIn( "fast", function() {

// console.log("MENU opacity 1.0");

});

break;

case "shop":

deselectItem("lessons");

selectItem(itemId);

$( "#lessonMenuDisplay" ).fadeOut( "fast", function() {

// console.log("MENU opacity 0");

});

$( "#lessonTextDisplay" ).fadeOut( "fast", function() {

// console.log("LESSON opacity 0");

});

$( "#storeroomMenuDisplay" ).fadeIn( "fast", function() {

// console.log("STOREROOM opacity 1.0");

});

break;

}

// == modify tab css between selected and active states

function selectItem(itemId) {

// console.log("selectItem");

var itemParentId = $('#' + itemId).parent('div').attr('id');

$('#' + itemParentId).removeClass('tab\_box\_active');

$('#' + itemParentId).addClass('tab\_box\_selected');

$('#' + itemId).removeClass('label\_text\_active');

$('#' + itemId).addClass('label\_text\_selected');

}

function deselectItem(itemId) {

// console.log("deselectItem");

var itemParentId = $('#' + itemId).parent('div').attr('id');

$('#' + itemParentId).removeClass('tab\_box\_selected');

$('#' + itemParentId).addClass('tab\_box\_active');

$('#' + itemId).removeClass('label\_text\_selected');

$('#' + itemId).addClass('label\_text\_active');

}

},

// ======= ======= ======= ACTIVATORS ======= ======= =======

// ======= ======= ======= ACTIVATORS ======= ======= =======

// ======= ======= ======= ACTIVATORS ======= ======= =======

// ======= activateLessonItems =======

activateLessonItems: function() {

console.log("activateLessonItems");

var page = this.activePage;

var setups = page.SetupItem;

var groups = page.GroupItems;

var gridders = page.GridItems;

var actors = page.ActorItems;

var targets = page.pageTargets;

var guides = page.guides;

var setupItem, setupTargets, setupControls;

var lessonItemsArray = [groups, gridders, actors, targets];

// == activate page level items

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

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

if ((lessonItemsArray[i]) && (lessonItemsArray[i].length > 0)) {

activatePageItems(lessonItemsArray[i]);

}

}

}

// == activate setup level items

if (setups.item) {

if (setups.targets) {

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

activatePageItems(setups.targets[i]);

}

}

if (setups.controls) {

console.log("setups.controls.length: ", setups.controls.length);

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

activatePageItems(setups.controls[i]);

}

}

}

// == match grid items (gridders) to frame indexes

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

gridders[i].indexedFrame = page.studio.indexedFrames[i];

}

// ======= activatePageItems =======

function activatePageItems(items) {

console.log("activatePageItems");

if (!items.length) {

console.log("\*\*\* !items.length \*\*\*");

items = [items];

}

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

var item = items[i];

// ======= MOUSEDOWN =======

$('#' + item.itemId).on('mousedown', function(e) {

console.log("\nmousedown");

var actor = clientApp.items[$(e.currentTarget).attr('id')];

var actorEl = $(e.currentTarget);

e.preventDefault();

clientApp.activeActor = actor;

actor.initMove(e, actorEl, actor);

});

// ======= MOUSEENTER/LEAVE =======

$('#' + item.itemId).on('mouseenter', function(e) {

// console.log("\nmouseenter");

clientApp.toggleHoverText(e.currentTarget, item.itemType);

});

$('#' + item.itemId).on('mouseleave', function(e) {

// console.log("\nmouseleave");

clientApp.toggleHoverText(null, null);

});

}

}

},

// ======= activateDisplayItems =======

activateDisplayItems: function() {

console.log("activateDisplayItems");

// == studio, shop, lesson select (CLICK)

$(".label\_text\_selected, .label\_text\_active").on('click', function(e) {

// console.log("\n-- click");

e.stopPropagation();

clientApp.selectSectionItem(e.currentTarget);

});

// == studio, shop, lesson hover text (ENTER/LEAVE)

$(".label\_text\_selected, .label\_text\_active").on('mouseenter', function(e) {

// console.log("\n-- mouseenter");

clientApp.toggleHoverText(e.currentTarget, "display");

e.stopPropagation();

});

$(".label\_text\_selected, .label\_text\_active").on('mouseleave', function(e) {

// console.log("\n-- mouseleave");

clientApp.toggleHoverText(null, null);

e.stopPropagation();

});

},

// ======= activateMenuItems =======

activateMenuItems: function(menu) {

console.log("activateMenuItems");

switch(menu) {

case "lessonMenu":

// == select lesson (CLICK)

$('#lessonMenu').children('li').children('div').on('click', function(e) {

console.log("\n-- click LESSON menu");

clientApp.activeLesson = clientApp.lessons[e.currentTarget.id];

clientApp.activePage = clientApp.pages["page\_" + clientApp.lessons[e.currentTarget.id].lessonIndex + "\_" + "0"];

clientApp.makeLessonPage(e.currentTarget);

e.stopPropagation();

});

// == lesson menu hover text

$('#lessonMenu').children('li').children('div').on('mouseenter', function(e) {

// console.log("\n-- mouseenter");

clientApp.activeLesson = e.currentTarget;

clientApp.toggleHoverText(e.currentTarget, "lesson");

e.stopPropagation();

});

$('#lessonMenu').children('li').children('div').on('mouseleave', function(e) {

// console.log("\n-- mouseleave");

clientApp.toggleHoverText(null, null);

e.stopPropagation();

});

break;

}

},

// ======= activatePrevNext =======

activatePrevNext: function() {

console.log("activatePrevNext");

$('#navPanel').children('div').on('click', function(e) {

console.log("\n -- click PREV/NEXT buttons");

var lessonPage = clientApp.getNextPage(e.currentTarget.id);

if ((lessonPage[0] != null) && (lessonPage[1] != null)) {

var nextLessonName = "lesson\_" + lessonPage[0];

var nextPageName = "page\_" + lessonPage[0] + "\_" + lessonPage[1];

if (clientApp.lessons[nextLessonName] && clientApp.pages[nextPageName]) {

clientApp.activeLesson = clientApp.lessons[nextLessonName];

clientApp.activePage = clientApp.pages[nextPageName];

if (clientApp.activeActor) {

clientApp.activeActor.dropLTWH = { L:0, T:0, W:0, H:0 };

clientApp.activeActor = null;

}

clientApp.makeLessonPage(e.currentTarget);

} else {

clientApp.updateLessonText("Sorry... requested page is missing. Click the <span class='hilight'>Lessons</span> tab to try again.");

}

} else {

clientApp.updateLessonText("Sorry... requested page is missing. Click the <span class='hilight'>Lessons</span> tab to try again.");

}

});

$('#navPanel').children('div').on('mouseenter', function(e) {

// console.log("-- mouseenter");

clientApp.toggleHoverText(e.currentTarget, null);

});

$('#navPanel').children('div').on('mouseleave', function(e) {

// console.log("-- mouseleave");

clientApp.toggleHoverText(null, null);

});

},

// ======= getNextPage =======

getNextPage: function(prevOrNext) {

console.log("getNextPage");

var lessonIndex = parseInt(clientApp.activeLesson.lessonIndex);

var lessonCount = \_.size(clientApp.lessons);

var pageIndex = parseInt(clientApp.activePage.pageKey.split("\_")[1]);

var pageCount = clientApp.activeLesson.pageKeys.length;

if (pageCount > 0) {

// == loop through pages of active lesson

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

var checkLessonIndex = parseInt(clientApp.activeLesson.pageKeys[i].split("\_")[0]);

var checkPageIndex = parseInt(clientApp.activeLesson.pageKeys[i].split("\_")[1]);

// == current page found

if ((lessonIndex == checkLessonIndex) && (pageIndex == checkPageIndex)) {

if (prevOrNext == "nextBtn") {

var nextPageIndex = parseInt(pageIndex + 1);

if (nextPageIndex >= pageCount) {

var nextLessonIndex = parseInt(lessonIndex + 1);

var nextPageIndex = 0;

if (nextLessonIndex == lessonCount) {

var nextLessonIndex = 0;

}

} else {

var nextLessonIndex = lessonIndex;

}

} else if (prevOrNext == "prevBtn") {

var nextPageIndex = parseInt(pageIndex - 1);

console.log("nextPageIndex:", nextPageIndex);

if (nextPageIndex < 0) {

var nextLessonIndex = parseInt(lessonIndex - 1);

console.log("nextLessonIndex:", nextLessonIndex);

if (nextLessonIndex >= 0) {

var lessonName = "lesson\_" + nextLessonIndex;

console.log("lessonName:", lessonName);

var nextPageIndex = parseInt(clientApp.lessons[lessonName] .pageKeys.length - 1);

} else {

var nextLessonIndex = 0;

}

} else {

var nextLessonIndex = lessonIndex;

}

}

return [nextLessonIndex, nextPageIndex]

}

}

}

},

// ======= activateMenuGrid =======

activateMenuGrid: function() {

console.log("activateMenuGrid");

},

// ======= ======= ======= UTILITIES ======= ======= =======

// ======= ======= ======= UTILITIES ======= ======= =======

// ======= ======= ======= UTILITIES ======= ======= =======

// ======= updateControlFrame =======

updateControlFrame: function(frameIndex) {

// console.log("updateControlFrame");

var item = clientApp.activeActor;

if (item.itemImage.image) {

urlString = "url('images/" + item.itemImage.image + "\_" + frameIndex + ".png') 0 0";

item.itemEl[0].style.background = urlString;

}

},

// ======= updateCanvasFrame =======

updateCanvasFrame: function(indexX, indexY) {

// console.log("updateCanvasFrame");

var page = clientApp.activePage;

// == get frame from nested array matrix (left/right, up/down) based on XY dragger indexes

if (clientApp.activePage.studio.image) {

if (clientApp.activePage.studio.matrix) {

if (page.studio.dir == "reverse") {

var studioImage = clientApp.studioImages[indexY][indexX];

} else {

var studioImage = clientApp.studioImages[indexX][indexY];

}

} else {

var studioImage = clientApp.studioImages[indexX];

}

var studioCan = this.displayItems["studio"].can;

var studioCtx = this.displayItems["studio"].ctx;

var canW = clientApp.displayItems["studio"].canW;

var canH = clientApp.displayItems["studio"].canH;

studioCtx.clearRect(0, 0, canW, canH);

if (studioImage) {

studioCtx.drawImage(studioImage, 0, 0, 720, 405, 0, 0, canW, canH);

studioCtx.save();

}

}

// == only some animations display on monitor canvas

if (clientApp.activePage.monitor.image) {

if (clientApp.activePage.monitor.matrix) {

if (page.monitor.dir == "reverse") {

var monitorImage = clientApp.monitorImages[indexY][indexX];

} else {

var monitorImage = clientApp.monitorImages[indexX][indexY];

}

} else {

if (clientApp.activePage.monitor.image) {

var monitorImage = clientApp.monitorImages[indexX];

}

}

var monitorCan = this.displayItems["monitor"].can;

var monitorCtx = this.displayItems["monitor"].ctx;

var canW = clientApp.displayItems["monitor"].canW;

var canH = clientApp.displayItems["monitor"].canH;

monitorCtx.clearRect(0, 0, canW, canH);

if (monitorImage) {

monitorCtx.drawImage(monitorImage, 0, 0, 720, 405, 0, 0, canW, canH);

studioCtx.save();

}

}

},

// ======= clearPageElements =======

clearPageElements: function() {

console.log("clearPageElements");

// == removeActorsGuides

$('#grid').empty();

$('#actors').empty();

$('#guides').empty();

$('#setup').empty();

$('#targets').empty();

$('.gridItem').remove();

$('.target').remove();

$('.control').remove();

$('.setupTarget').remove();

},

// ======= clearLessonCanvases =======

clearLessonCanvases: function() {

console.log("clearLessonCanvases");

var canvases = ["studio", "monitor"];

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

this.displayItems[canvases[i]].ctx.clearRect(0, 0, 720, 405);

}

},

// ======= toggleHoverText =======

toggleHoverText: function(item, itemType) {

// console.log("toggleHoverText");

if (clientApp.displayItems.warningFlag == false) {

if ($(item).attr('id')) {

if (itemType == "display") {

var itemText = clientApp.displayItems[$(item).attr('id')].itemText;

} else if (itemType == "lesson") {

var itemText = clientApp.lessons[$(item).attr('id')].itemText;

} else if ((itemType == "actor") || (itemType == "gridItem")) {

var itemText = clientApp.items[$(item).attr('id')].itemText;

} else if (itemType == "setupTarget") {

var target = clientApp.targets[$(item).attr('id')]

var itemText = target.itemText + target.itemName;

} else {

var itemText = $(item).attr('id');

}

$('#hover\_text').text(itemText);

} else {

$('#hover\_text').text('');

}

}

}

};

// == error-box functions

$('#close-error, #close-message').on('click', function(e) {

e.stopPropagation();

$('#error-box, #message-box').css('display', 'none');

});

clientApp.initialize();

// ======= ======= ======= ======= ======= behaviors.js ======= ======= ======= ======= =======

// ======= ======= ======= ======= ======= behaviors.js ======= ======= ======= ======= =======

// ======= ======= ======= ======= ======= behaviors.js ======= ======= ======= ======= =======

// ======= ======= ======= INIT MOVE ======= ======= =======

// ======= ======= ======= INIT MOVE ======= ======= =======

// ======= ======= ======= INIT MOVE ======= ======= =======

// ======= initMove =======

Item.prototype.initMove = function(e) {

console.log("initMove");

var page = clientApp.activePage;

// == get active item element

var actor = clientApp.activeActor;

var itemType = actor.itemType;

actor.itemEl = $("#" + actor.itemId).eq(0);

// == strip "px" suffix from left and top properties

var locL = parseInt($(actor.itemEl).css('left').substring(0, $(actor.itemEl).css('left').length - 2));

var locT = parseInt($(actor.itemEl).css('top').substring(0, $(actor.itemEl).css('top').length - 2));

// == limit moves to LTWH (left/top/width/height) boundaries

if (actor.bounds.W > (displayItems.studio.canW - actor.initLTWH.W)) {

var itemBoundsW = displayItems.studio.canW - actor.initLTWH.W;

} else {

var itemBoundsW = actor.bounds.W;

}

if (actor.bounds.H > (displayItems.studio.canH - actor.initLTWH.H)) {

var itemBoundsH = displayItems.studio.canH - actor.initLTWH.H;

} else {

var itemBoundsH = actor.bounds.H;

}

// ======= ======= ACTOR ======= =======

// ======= ======= ACTOR ======= =======

// ======= ======= ACTOR ======= =======

// == set item/mouse locations and bounds (absolute position)

actor.startXY.itemL = locL;

actor.startXY.itemT = locT;

actor.startXY.mouseX = e.clientX;

actor.startXY.mouseY = e.clientY;

actor.startXY.diffX = e.clientX - locL;

actor.startXY.diffY = e.clientY - locT;

actor.minMaxLT.minL = displayItems.studio.canL + actor.bounds.L;

actor.minMaxLT.minT = displayItems.studio.canT + actor.bounds.T;

actor.minMaxLT.maxL = displayItems.studio.canL + actor.bounds.L + itemBoundsW;

actor.minMaxLT.maxT = displayItems.studio.canT + actor.bounds.T + itemBoundsH;

// ======= ======= TARGETS ======= =======

// ======= ======= TARGETS ======= =======

// ======= ======= TARGETS ======= =======

// == page has active targets

if (page.pageTargets.length > 0) {

var pageTargets = page.pageTargets;

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

var target = pageTargets[i];

target.absLoc.L = displayItems.studio.canL + target.initLTWH.L;

target.absLoc.T = displayItems.studio.canT + target.initLTWH.T;

target.absLoc.W = displayItems.studio.canL + target.initLTWH.L + target.initLTWH.W;

target.absLoc.H = displayItems.studio.canT + target.initLTWH.T + target.initLTWH.H;

}

}

// == setup item has active targets (and may have other non-active targets)

if (page.SetupItem.targets.length > 0) {

var setupItem = page.SetupItem.item;

for (var i = 0; i < page.SetupItem.targets.length; i++) {

var target = page.SetupItem.targets[i];

target.absLoc.L = displayItems.studio.canL + setupItem.initLTWH.L + target.initLTWH.L;

target.absLoc.T = displayItems.studio.canT + setupItem.initLTWH.T + target.initLTWH.T;

target.absLoc.W = displayItems.studio.canL + setupItem.initLTWH.L + target.initLTWH.L + target.initLTWH.W;

target.absLoc.H = displayItems.studio.canT + setupItem.initLTWH.T + target.initLTWH.T + target.initLTWH.H;

}

}

// ======= ======= CONTROLS ======= =======

// ======= ======= CONTROLS ======= =======

// ======= ======= CONTROLS ======= =======

// == over-ride previous bounds values for control actors (they do not move; swap frames only)

if (page.SetupItem.controls.length > 0) {

var setupItem = page.SetupItem.item;

var control = page.SetupItem.controls[0];

actor.minMaxLT.minL = displayItems.studio.canL + setupItem.initLTWH.L + control.initLTWH.L + actor.bounds.L;

actor.minMaxLT.minT = displayItems.studio.canT + setupItem.initLTWH.T + control.initLTWH.T + actor.bounds.T;

actor.minMaxLT.maxL = displayItems.studio.canL + setupItem.initLTWH.L + control.initLTWH.L + actor.bounds.L + itemBoundsW;

actor.minMaxLT.maxT = displayItems.studio.canT + setupItem.initLTWH.T + control.initLTWH.T + actor.bounds.T + itemBoundsH;

}

window.addEventListener('mousemove', actor.moveItem, true);

window.addEventListener('mouseup', actor.mouseUp, true);

}

// ======= ======= ======= MOUSE MOVE ======= ======= =======

// ======= ======= ======= MOUSE MOVE ======= ======= =======

// ======= ======= ======= MOUSE MOVE ======= ======= =======

// ======= moveItem =======

Item.prototype.moveItem = function(e) {

// console.log("moveItem");

var page = clientApp.activePage;

var actor = clientApp.activeActor;

var itemMove = actor.itemMove;

// == calculate change in mouse X/Y location in pixels

var dX = parseInt(e.clientX - actor.startXY.mouseX);

var dY = parseInt(e.clientY - actor.startXY.mouseY);

// == calculate percentage of X/Y travel across item bounds area

var deltaX = ((dX + actor.dropLTWH.L)/actor.bounds.W).toFixed(2);

var deltaY = ((dY + actor.dropLTWH.T)/actor.bounds.H).toFixed(2);

switch(itemMove) {

case "control":

var left = parseInt(actor.startXY.itemL + dX);

var top = parseInt(actor.startXY.itemT + dY);

var itemLT = getMoveBoundaries(left, top);

getControlFrames(itemLT[0], itemLT[1], deltaX, deltaY);

break;

case "matrixAB":

var left = parseInt(actor.startXY.itemL + dX);

var top = parseInt(actor.startXY.itemT + dY);

var itemLT = getMoveBoundaries(left, top);

updateMatrixAB(itemLT[0], itemLT[1]);

break;

case "dragger":

var left = parseInt(actor.startXY.itemL + dX);

var top = parseInt(actor.startXY.itemT + dY);

var itemLT = getMoveBoundaries(left, top);

// == check for setup or page targets

if (page.SetupItem.targets.length > 0) {

checkItemTargets(itemLT[0], itemLT[1], "setup");

} else if (page.pageTargets.length > 0) {

checkItemTargets(itemLT[0], itemLT[1], "page");

} else {

updateItemLoc(itemLT[0], itemLT[1]);

}

break;

case "slider":

var left = parseInt(actor.startXY.itemL + dX);

var top = parseInt(actor.startXY.itemT - actor.dropLTWH.T - (actor.bounds.H \* deltaX));

var itemLT = getMoveBoundaries(left, top);

updateItemLoc(itemLT[0], itemLT[1]);

break;

}

// ======= updateMatrixAB =======

function updateMatrixAB(left, top) {

// console.log("updateMatrixAB");

// == calculate percent movement through frameset/limit frames to start/end

var indexX = Math.round(-deltaX \* page.studio.endFrame);

if (indexX < 0) {

indexX = 0;

}

if (indexX > page.studio.endFrame) {

indexX = page.studio.endFrame;

}

var indexY = Math.round(deltaY \* page.studio.endFrame);

if (indexY < 0) {

indexY = 0;

}

if (indexY > page.studio.endFrame) {

indexY = page.studio.endFrame;

}

// == set real-time item loc based on slider position

$(actor.itemEl).css('z-index', '10');

$(actor.itemEl).css('top', top + 'px');

$(actor.itemEl).css('left', left + 'px');

actor.startXY.dragL = left;

actor.startXY.dragT = top;

// == set real-time canvas frame based on slider position

clientApp.updateCanvasFrame(indexX, indexY);

// == update locXY indicator

$('#locXYWH').html("<p class='info-text'>left: " + left + "</p><p class='info-text'>top: " + top + "</p>");

}

// ======= checkItemTargets =======

function checkItemTargets(left, top, targetType) {

// console.log("checkItemTargets");

if (targetType == "page") {

var targetList = page.pageTargets;

} else if (targetType == "setup") {

var targetList = page.SetupItem.targets;

}

// == set real-time item loc and canvas frame based on slider position

$(actor.itemEl).css('z-index', '10');

$(actor.itemEl).css('top', top + 'px');

$(actor.itemEl).css('left', left + 'px');

actor.startXY.dragL = left;

actor.startXY.dragT = top;

// == init collision detector for target

var target;

var draggerL = left + actor.initLTWH.W/2;

var draggerT = top + actor.initLTWH.H/2;

// == search available targets for collision

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

target = targetList[i];

// == COLLISION with target

if ((draggerL < target.absLoc.W) && (draggerT > target.absLoc.T) && (draggerL > target.absLoc.L) && (draggerT < target.absLoc.H)) {

console.log("-- HIT -- HIT -- HIT --");

// == locate dragged item at top/left of target

$(actor.itemEl).off();

$(actor.itemEl).css('top', target.absLoc.T + 'px');

$(actor.itemEl).css('left', target.absLoc.L + 'px');

$(actor.itemEl).css('background-size', target.initLTWH.W + 'px');

$(actor.itemEl).css('background-repeat', 'no-repeat');

$(actor.itemEl).css('width', target.initLTWH.W + 'px');

$(actor.itemEl).css('height', target.initLTWH.H + 'px');

clientApp.updateCanvasFrame(actor.indexedFrame, null);

swapTargetOccupiers(target, actor);

window.removeEventListener('mousemove', actor.moveItem, true);

window.removeEventListener('mouseup', actor.mouseUp, true);

}

}

}

// ======= swapTargetOccupiers =======

function swapTargetOccupiers(target, newOccupier) {

console.log("swapTargetOccupiers");

// == fade out new occupier

$(newOccupier.itemEl).fadeOut(1000, function() {

// == return target occupier to its original location

var occupier = target.occupier;

if (occupier) {

$(occupier.itemEl).css('visibility', 'visible');

$(occupier.itemEl).css('display', 'block');

$(occupier.itemEl).css('width', target.occupier.initLTWH.W \* 0.4);

$(occupier.itemEl).css('height', target.occupier.initLTWH.H \* 0.4);

$(occupier.itemEl).css('z-index', '10');

$(occupier.itemEl).css('background-size', '100%');

$(occupier.itemEl).animate({

width: occupier.initLTWH.W,

height: occupier.initLTWH.H,

left: occupier.initLTWH.L,

top: occupier.initLTWH.T

}, 500, function() {

console.log("itemReturned");

// == restore event listener on previous target occupier

$(occupier.itemEl).on('mousedown', function(e) {

console.log("\nmousedown");

window.removeEventListener('mouseup', actor.mouseUp, true);

var actor = clientApp.items[$(e.currentTarget).attr('id')];

var actorEl = $(e.currentTarget);

e.preventDefault();

clientApp.activeActor = actor;

actor.initMove(e, actorEl, actor);

});

});

}

// == install newOccupier on target

target.occupier = newOccupier;

});

}

// ======= getControlFrames =======

function getControlFrames(dX, dY, deltaX, deltaY) {

// console.log("getControlFrames");

// == calculate percent movement through frameset/limit frames to start/end

var frameIndex = Math.round(-deltaX \* actor.itemImage.endFrame);

if (frameIndex < 0) {

frameIndex = 0;

}

if (frameIndex > actor.itemImage.endFrame) {

frameIndex = actor.itemImage.endFrame;

}

clientApp.updateControlFrame(frameIndex, null);

clientApp.updateCanvasFrame(frameIndex, null);

}

// ======= updateItemLoc =======

function updateItemLoc(left, top) {

// console.log("updateItemLoc");

// == calculate percent movement through frameset/limit frames to start/end

var frameIndex = Math.round(-deltaX \* clientApp.activePage.studio.endFrame);

if (frameIndex < 0) {

frameIndex = 0;

}

if (frameIndex > clientApp.activePage.studio.endFrame) {

frameIndex = clientApp.activePage.studio.endFrame;

}

// == set real-time item loc based on slider position

$(actor.itemEl).css('z-index', '10');

$(actor.itemEl).css('top', top + 'px');

$(actor.itemEl).css('left', left + 'px');

actor.startXY.dragL = left;

actor.startXY.dragT = top;

// == update screen XY locator (for development)

$('#locXYWH').html("<p class='info-text'>left: " + left + "</p><p class='info-text'>top: " + top + "</p>");

// == set real-time canvas frame based on slider/dragger position

clientApp.updateCanvasFrame(frameIndex, null);

}

// ======= ======= ======= MATH ======= ======= =======

// ======= ======= ======= MATH ======= ======= =======

// ======= ======= ======= MATH ======= ======= =======

// ======= getMoveBoundaries =======

function getMoveBoundaries(left, top) {

// console.log("getMoveBoundaries");

if (left < actor.minMaxLT.minL) {

left = actor.minMaxLT.minL;

}

if (left > actor.minMaxLT.maxL) {

left = actor.minMaxLT.maxL;

}

if (top < actor.minMaxLT.minT) {

top = actor.minMaxLT.minT;

}

if (top > actor.minMaxLT.maxT) {

top = actor.minMaxLT.maxT;

}

return [left, top];

}

}

// ======= ======= ======= MOUSE UP ======= ======= =======

// ======= ======= ======= MOUSE UP ======= ======= =======

// ======= ======= ======= MOUSE UP ======= ======= =======

// ======= mouseUp =======

Item.prototype.mouseUp = function(e) {

console.log("mouseUp");

var actor = clientApp.activeActor;

$(clientApp.activeActor.itemEl).off();

window.removeEventListener('mousemove', actor.moveItem, true);

// == store relative loc where item was dropped

actor.dropLTWH.L = actor.startXY.dragL - (clientApp.displayItems.studio.canL + actor.bounds.L + actor.bounds.W);

actor.dropLTWH.T = actor.startXY.dragT - (clientApp.displayItems.studio.canT + actor.bounds.T);

actor.dropLTWH.W = null;

actor.dropLTWH.H = null;

// == reactivate item

clientApp.activateLessonItems();

}

// ======= ======= ======= ======= ======= lessonObjects.js ======= ======= ======= ======= =======

// ======= ======= ======= ======= ======= lessonObjects.js ======= ======= ======= ======= =======

// ======= ======= ======= ======= ======= lessonObjects.js ======= ======= ======= ======= =======

var defaultLesson = {

lessonIndex: 0,

lessonTitle: "lessonTitle",

lessonSubtitle: "lessonSubtitle",

pageKeys: []

}

function initLessons() {

console.log("initLessons");

// lessonTitle, lessonSubtitle

var lessons = {};

// ======= demo items =======

lessons.lesson\_0 = new Lesson (

/\* lessonIndex \*/ 0,

/\* lessonTitle \*/ "Intro to Lighting",

/\* lessonSubtitle \*/ "qualtity, quality, color position",

/\* pageKeys \*/ ["0\_0"]

);

lessons.lesson\_1 = new Lesson (

/\* lessonIndex \*/ 1,

/\* lessonTitle \*/ "Intensity",

/\* lessonSubtitle \*/ "lights and brightness",

/\* pageKeys \*/ ["1\_0", "1\_1", "1\_2", "1\_3"]

);

lessons.lesson\_2 = new Lesson (

/\* lessonIndex \*/ 2,

/\* lessonTitle \*/ "Qualtity",

/\* lessonSubtitle \*/ "soft light / hard light",

/\* pageKeys \*/ ["2\_0", "2\_1", "2\_2", "2\_3", "2\_4", "2\_5"]

);

...

return lessons;

}

function Lesson (lessonIndex, lessonTitle, lessonSubtitle, pageKeys) {

// console.log(' Lesson');

this.lessonIndex = lessonIndex;

this.lessonTitle = lessonTitle;

this.lessonSubtitle = lessonSubtitle;

this.pageKeys = pageKeys;

}

// ======= ======= ======= ======= ======= pageObjects.js ======= ======= ======= ======= =======

// ======= ======= ======= ======= ======= pageObjects.js ======= ======= ======= ======= =======

// ======= ======= ======= ======= ======= pageObjects.js ======= ======= ======= ======= =======

function initPages(items, targets) {

console.log("initPages");

// pageKey, pageText, SetupItem, GroupItems, GridItems, ActorItems, pageTargets, studioCanvas, monitorCanvas

var pages = {};

...

// ======= ======= ======= ======= ======= intensity ======= ======= ======= ======= =======

pages.page\_1\_2 = new Page (

/\* pageKey \*/ "1\_2",

/\* pageText \*/ "Often, moving lights is not easy -- especially when its a 20k HMI lighting up a city block. But if it's too bright you can drop a <span class='hilight'>SCRIM</span> into the <span class='hilight'>barn door frame</span> and cut it down a bit. <span class='hilightY'>Drag the scrims</span> towards the light and drop them into the barndoor frame. Easy!",

/\* SetupItem \*/ { item:items.f650\_0\_2, targets:[targets.barndoorFrame], controls:[] },

/\* GroupItems \*/ [],

/\* GridItems \*/ [items.scrim1\_0, items.scrim2\_0, items.scrim3\_0],

/\* ActorItems \*/ [],

/\* pageTargets \*/ [],

/\* guides \*/ [],

/\* studio \*/ { folder:null, image:"st\_int\_power", startFrame:0, endFrame:4, initFrame:4, indexedFrames:[3, 2, 1] },

/\* monitor \*/ { folder:null, image:"mn\_int\_power", startFrame:0, endFrame:4, initFrame:4, indexedFrames:[3, 2, 1] }

);

...

function Page (pageKey, pageText, SetupItem, GroupItems, GridItems, ActorItems, pageTargets, guides, studio, monitor) {

// console.log(' Page');

this.pageKey = pageKey;

this.pageText = pageText;

this.SetupItem = SetupItem;

this.GroupItems = GroupItems;

this.GridItems = GridItems;

this.ActorItems = ActorItems;

this.pageTargets = pageTargets;

this.guides = guides;

this.studio = studio;

this.monitor = monitor;

}

return pages;

}

// ======= ======= ======= ======= ======= itemObjects.js ======= ======= ======= ======= =======

// ======= ======= ======= ======= ======= itemObjects.js ======= ======= ======= ======= =======

// ======= ======= ======= ======= ======= itemObjects.js ======= ======= ======= ======= =======

function initItems(targets) {

console.log("initItems");

var items = {};

var canvasS = clientApp.displayItems.studio;

...

items.f650\_0\_2 = new Item(

/\* itemId \*/ "f650\_0\_2",

/\* itemEl \*/ null,

/\* itemName \*/ "scrims\_650",

/\* itemText \*/ "",

/\* itemType \*/ "setup",

/\* itemMove \*/ "none",

/\* itemImage \*/ "f650\_0",

/\* itemTargets \*/ [targets.barndoorFrame],

/\* itemControls \*/ [items.spotFlood],

/\* startXY \*/ { itemL:0, itemT:0, mouseX:0, mouseY:0, diffX:0, diffY:0, dragL:0, dragT:0 },

/\* minMaxLT \*/ { minL:0, minT:0, maxL:0, maxT:0 },

/\* dropLTWH \*/ { L:0, T:0, W:0, H:0 },

/\* initLTWH \*/ { L:400, T:80, W:350, H:270 },

/\* bounds \*/ { L:400, T:80, W:350, H:270 }

);

...

return items;

}

function Item(itemId, itemEl, itemName, itemText, itemType, itemMove, itemImage, itemTargets, itemControls, startXY, minMaxLT, dropLTWH, initLTWH, bounds) {

this.itemId = itemId;

this.itemEl = itemEl;

this.itemName = itemName;

this.itemText = itemText;

this.itemType = itemType;

this.itemMove = itemMove;

this.itemImage = itemImage;

this.itemTargets = itemTargets;

this.itemControls = itemControls;

this.startXY = startXY;

this.minMaxLT = minMaxLT;

this.initLTWH = initLTWH;

this.dropLTWH = dropLTWH;

this.bounds = bounds;

}