Exercise 11.22

Table

Description automatically generated with medium confidence

function main()

{

dateInput = document.getElementById("dateInput");

submitButton = document.getElementById("submitButton");

submitButton.addEventListener("click",function(){

var date = new Date(dateInput.value);

var getDate=document.getElementById("getDate");

getDate.innerHTML = date.getDate();

var getUTCDate=document.getElementById("getUTCDate");

getUTCDate.innerHTML = date.getUTCDate();

var getDay=document.getElementById("getDay");

getDay.innerHTML = date.getDay();

var getFullYear=document.getElementById("getFullYear");

getFullYear.innerHTML = date.getFullYear();

var getUTCFullYear=document.getElementById("getUTCFullYear");

getUTCFullYear.innerHTML = date.getUTCFullYear();

var getHours=document.getElementById("getHours");

getHours.innerHTML = date.getHours();

var getUTCHours=document.getElementById("getUTCHours");

getUTCHours.innerHTML = date.getUTCHours();

var getMilliseconds=document.getElementById("getMilliseconds");

getMilliseconds.innerHTML = date.getMilliseconds();

var getUTCMilliseconds=document.getElementById("getUTCMilliseconds");

getUTCMilliseconds.innerHTML = date.getUTCMilliseconds();

var getMinutes=document.getElementById("getMinutes");

getMinutes.innerHTML = date.getMinutes();

var getUTCMinutes=document.getElementById("getUTCMinutes");

getUTCMinutes.innerHTML = date.getUTCMinutes();

var getMonth=document.getElementById("getMonth");

getMonth.innerHTML = date.getMonth();

var getUTCMonth=document.getElementById("getUTCMonth");

getUTCMonth.innerHTML = date.getUTCMonth();

var getSeconds=document.getElementById("getSeconds");

getSeconds.innerHTML = date.getSeconds();

var getUTCSeconds=document.getElementById("getUTCSeconds");

getUTCSeconds.innerHTML = date.getUTCSeconds();

})

};

Exercise 12.5

Graphical user interface, application, Word

Description automatically generated

// GLOBALS

var countVal = 0;

// increments the disabled input to the count value

function count(){

var counter = document.getElementById("counter");

var count = document.getElementById("count");

counter.addEventListener("click", function(){

count.value = ++countVal;

})

}

// initial entry point

function main(){

count();

}

Exercise 13.3

The picture for this exercise doesn’t display the functionality of the program. The canvas has a pencil tool enabled by default. If the user presses the alt key then the eraser function is enabled. If they wish to toggle back to the pencil they just need to press alt again.

A picture containing linedrawing

Description automatically generated

// Keep everything in anonymous function, called on window load.

if(window.addEventListener) {

window.addEventListener('load', function () {

var canvas, context, tool;

function init () {

// Find the canvas element.

canvas = document.getElementById('imageView');

if (!canvas) {

alert('Error: I cannot find the canvas element!');

return;

}

if (!canvas.getContext) {

alert('Error: no canvas.getContext!');

return;

}

// Get the 2D canvas context.

context = canvas.getContext('2d');

if (!context) {

alert('Error: failed to getContext!');

return;

}

// Pencil tool instance.

tool = new tool\_pencil();

// Attach the mousedown, mousemove and mouseup event listeners.

canvas.addEventListener('mousedown', ev\_canvas, false);

canvas.addEventListener('mousemove', ev\_canvas, false);

canvas.addEventListener('mouseup', ev\_canvas, false);

}

// This painting tool works like a drawing pencil which tracks the mouse

// movements.

function tool\_pencil () {

var tool = this;

this.started = false;

// This is called when you start holding down the mouse button.

// This starts the pencil drawing.

this.mousedown = function (ev) {

context.beginPath();

context.moveTo(ev.\_x, ev.\_y);

tool.started = true;

};

// This function is called every time you move the mouse. Obviously, it only

// draws if the tool.started state is set to true (when you are holding down

// the mouse button).

this.mousemove = function (ev) {

if (tool.started) {

context.lineTo(ev.\_x, ev.\_y);

context.strokeStyle = "#000000";

context.stroke();

}

};

// This is called when you release the mouse button.

this.mouseup = function (ev) {

if (tool.started) {

tool.mousemove(ev);

tool.started = false;

}

};

}

// This painting tool works like a drawing pencil which tracks the mouse

// movements.

function tool\_eraser () {

var tool = this;

this.started = false;

// This is called when you start holding down the mouse button.

// This starts the pencil drawing.

this.mousedown = function (ev) {

context.beginPath();

context.moveTo(ev.\_x, ev.\_y);

tool.started = true;

};

// This function is called every time you move the mouse. Obviously, it only

// draws if the tool.started state is set to true (when you are holding down

// the mouse button).

this.mousemove = function (ev) {

if (tool.started) {

context.lineTo(ev.\_x, ev.\_y);

context.strokeStyle = "#FFFFFF";

context.stroke();

}

};

// This is called when you release the mouse button.

this.mouseup = function (ev) {

if (tool.started) {

tool.mousemove(ev);

tool.started = false;

}

};

}

// The general-purpose event handler. This function just determines the mouse

// position relative to the canvas element.

function ev\_canvas (ev) {

if (ev.layerX || ev.layerX == 0) { // Firefox

ev.\_x = ev.layerX;

ev.\_y = ev.layerY;

} else if (ev.offsetX || ev.offsetX == 0) { // Opera

ev.\_x = ev.offsetX;

ev.\_y = ev.offsetY;

}

// Call the event handler of the tool.

var func = tool[ev.type];

if (func) {

func(ev);

}

}

init();

var isAltPressed = false;

document.addEventListener("keydown",function(e){

if(e.key == "Alt"){

if(isAltPressed){

tool = new tool\_pencil();

console.log(isAltPressed);

isAltPressed = false;

}

else{

tool = new tool\_eraser();

console.log(isAltPressed);

isAltPressed = true;

}

}

});

}, false); }

Exercise 13.5

When a user types an ‘&’ in the text field it will show up as ‘&’. If they click off of the field triggering a blur event the text will convert any ‘&’ into ‘and’s.

Graphical user interface

Description automatically generated with medium confidence

//Initial Entry point

function main(){

var formElement = document.getElementById("amepersand");

formElement.addEventListener("blur",function(e){

var text = formElement.value;

text = text.replace("&","and");

formElement.value = text;

})

}

#3 HTML canvas exercise

A picture containing letter

Description automatically generated

// Keep everything in anonymous function, called on window load.

if(window.addEventListener) {

window.addEventListener('load', function () {

var canvas, context, tool;

function init () {

// Find the canvas element.

canvas = document.getElementById('imageView');

if (!canvas) {

alert('Error: I cannot find the canvas element!');

return;

}

if (!canvas.getContext) {

alert('Error: no canvas.getContext!');

return;

}

// Get the 2D canvas context.

context = canvas.getContext('2d');

if (!context) {

alert('Error: failed to getContext!');

return;

}

// Pencil tool instance.

tool = new tool\_pencil();

// Attach the mousedown, mousemove and mouseup event listeners.

canvas.addEventListener('mousedown', ev\_canvas, false);

canvas.addEventListener('mousemove', ev\_canvas, false);

canvas.addEventListener('mouseup', ev\_canvas, false);

}

// This painting tool works like a drawing pencil which tracks the mouse

// movements.

function tool\_pencil () {

var tool = this;

this.started = false;

// This is called when you start holding down the mouse button.

// This starts the pencil drawing.

this.mousedown = function (ev) {

context.beginPath();

context.moveTo(ev.\_x, ev.\_y);

tool.started = true;

};

// This function is called every time you move the mouse. Obviously, it only

// draws if the tool.started state is set to true (when you are holding down

// the mouse button).

this.mousemove = function (ev) {

if (tool.started) {

context.lineTo(ev.\_x, ev.\_y);

context.stroke();

}

};

// This is called when you release the mouse button.

this.mouseup = function (ev) {

if (tool.started) {

tool.mousemove(ev);

tool.started = false;

}

};

}

// This painting tool works like a drawing pencil which tracks the mouse

// movements.

function tool\_eraser () {

var tool = this;

this.started = false;

// This is called when you start holding down the mouse button.

// This starts the pencil drawing.

this.mousedown = function (ev) {

context.beginPath();

context.moveTo(ev.\_x, ev.\_y);

tool.started = true;

};

// This function is called every time you move the mouse. Obviously, it only

// draws if the tool.started state is set to true (when you are holding down

// the mouse button).

this.mousemove = function (ev) {

if (tool.started) {

context.lineTo(ev.\_x, ev.\_y);

context.strokeStyle = "#FFFFFF";

context.stroke();

}

};

// This is called when you release the mouse button.

this.mouseup = function (ev) {

if (tool.started) {

tool.mousemove(ev);

tool.started = false;

}

};

}

// The general-purpose event handler. This function just determines the mouse

// position relative to the canvas element.

function ev\_canvas (ev) {

if (ev.layerX || ev.layerX == 0) { // Firefox

ev.\_x = ev.layerX;

ev.\_y = ev.layerY;

} else if (ev.offsetX || ev.offsetX == 0) { // Opera

ev.\_x = ev.offsetX;

ev.\_y = ev.offsetY;

}

// Call the event handler of the tool.

var func = tool[ev.type];

if (func) {

func(ev);

}

}

init();

var eraser = document.getElementById("eraser");

eraser.addEventListener("click",function(){

tool = new tool\_eraser();

})

var pencil = document.getElementById("pencil");

pencil.addEventListener("click",function(){

tool = new tool\_pencil();

})

var clear = document.getElementById("clear");

clear.addEventListener("click", function(){

context.clearRect(0,0,canvas.width,canvas.height);

})

}, false); }