*Practical Lab Exercises*

Lab - Javascript Prt 2

Web Programming (F28WP)

# Introduction

In this lab, you’ll further develop your understanding of Javascript.

### 1.1 Features/Experiment

Take the `basic’ skeleton Javascript code for Santa game:

[<LINK>](https://f28wp.github.io/material/labs/javascriptsanta/JavaScriptSanta.html)

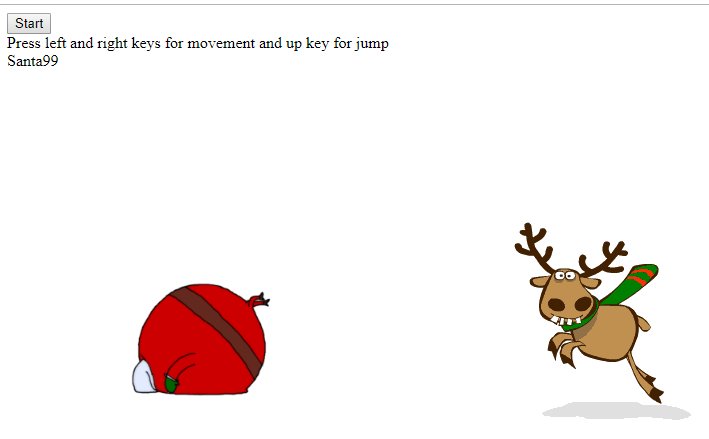
Add additional features (using Javascript).

For example:

* Collision detection
* Score/pause
* Improved GUI/Layout
* Animations
* Multiplayer
* …

Note. The Javascript Santa game is currently in a single .html file, with two images (img.gif and ball.gif)

Zip file with the html and images [<LINK>](https://f28wp.github.io/material/labs/javascriptsanta/javascriptsanta.zip)



### 2.0 Drawing Program

Create a new html file and add the following script. The script demonstrates a `minimal’ drawing program using the mouse. So you can drag the mouse cursor around the screen to draw. Enhance the implementation.

// create canvas element and append it to document body

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

document.body.appendChild(canvas);

// some hotfixes

document.body.style.margin = 0;

canvas.style.position = 'fixed';

// get canvas 2D context and set him correct size

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

resize();

// last known position

var pos = { x: 0, y: 0 };

window.addEventListener('resize', resize);

document.addEventListener('mousemove', draw);

document.addEventListener('mousedown', setPosition);

document.addEventListener('mouseenter', setPosition);

// new position from mouse event

function setPosition(e) {

pos.x = e.clientX;

pos.y = e.clientY;

}

// resize canvas

function resize() {

ctx.canvas.width = window.innerWidth;

ctx.canvas.height = window.innerHeight;

}

function draw(e) {

// mouse left button must be pressed

if (e.buttons !== 1) return;

ctx.beginPath(); // begin

ctx.lineWidth = 5;

ctx.lineCap = 'round';

ctx.strokeStyle = '#c0392b';

ctx.moveTo(pos.x, pos.y); // from

setPosition(e);

ctx.lineTo(pos.x, pos.y); // to

ctx.stroke(); // draw it!

}

Task 1. Implement the minimum working example and ensure it works

Task 2. Add additional features, for example, `clear` button

Task 3. Add a `save’ button, so you store the drawing (e.g., use an array to store the positions as the drawing evolves). Then when clear is pressed, you have a `restore’ button to draw the saved image back to the screen.

Task 4. Add a colour pick option button (so you can select a colour to be drawn on the screen.

Task 5. Add a line thickness button (so you can choose how thick the line is that you’ll draw with)