CSCU9A2 **Spring 2017**

Web Design Practical 4: HTML5 video and animation

AIMS : INTRODUCE USING VIDEOS AND CREATING ANIMATIONS WITH HTML5.

This sheet contains <u>one</u> checkpoint. Deadline: Thursday 6th April

FURTHER DOCUMENTATION

There is a brief overview of the HTML5 canvas in the HTML reference document on the CSCU9A2 website. For more detail, consult *HTML5: the Missing Manual*, available as an e-book from the library. See also the W3Schools tutorial on HTML5 at `http://www.w3schools.com/html/html5_intro.asp`.

THE HTML5 VIDEO TAG

HTML5 is becoming the dominant HTML standard on the Web. It is in many ways simpler to use than earlier standards. It also incorporates features that make it possible to include audio, video and animation directly within an HTML document, removing the need for external plug-ins like Flash. We will take a brief look at some of these features in this practical.

Copy the folder `My Computer\Groups (V:)\HTML5` into your Web folder. Navigate into this folder. View the document `animations.html` within a web browser.

Next, open `animations.html` in TextPad. This document contains an example of the `<video>` tag in HTML5. Some points to note:

- The `controls` attribute instructs the web browser to add basic video controls like play, pause, and volume (though this example has no sound). JavaScript can be used to add more controls – see later.

- Using video in web pages can create problems with accessibility as videos cannot be interpreted by screen reading software. Unlike the `<img>` tag, the `<video>` tag does not support the `alt` attribute for providing alternative text. It is possible to insert text content between the opening and closing `<video>` tags for browser that do not support this tag, and this can be used to provide a brief textual description of the video. For longer videos it is recommended to include a separate textual transcript.

JavaScript can be used to manipulate a video element in a web page, just as it can be used with other elements. Use your knowledge of JavaScript (refer back to the last practical if necessary) to add a button to the page which toggles the video between small and big sizes.

Follow these steps:

- Give the video element an id, say `'illusion'`, so that it can be retrieved by id.
- Add a button labelled “big / small” in a suitable place.
- Write a JavaScript function called `toggleSize` which resizes the video by modifying its width and height attributes. Note that you will need to use a variable to keep track of whether the video is currently “big” or “small”.
- Add code to call `toggleSize` when the button is clicked.

THE HTML5 CANVAS AND ANIMATION

The HTML5 `<canvas>` element can be used to create graphical animations on the fly using a scripting language such as JavaScript. Before HTML5 animations could only be created using external plug-ins such as Flash.

HTML5 is fun! Before we go further, take a look at some of the examples listed on this site:

`webdesignledger.com/inspiration/12-fun-clever-examples-of-html5`.

The basis of animation in HTML5 is the `<canvas>` element. This provides a rectangular area on the screen which acts as a container for graphics. Add a canvas element to your document:

```
<canvas id="myCanvas" width="500" height="500">
  Your browser does not support the HTML5 canvas tag.
</canvas>
```

Drawing on the canvas is done by JavaScript. The script below first retrieves the canvas element you just created (using `getElementById`). The canvas is equipped with a powerful set of commands for drawing lines, shapes, and text and other graphics. These commands are packaged up in the canvas’s *2D drawing context object*. The second line of the script retrieves this object and stores it in a variable called `context`.

Add this script to your document:

```
<script>
   var canvas = document.getElementById("myCanvas");
   var context = canvas.getContext("2d");
</script>
```

Think of the canvas as a two-dimensional grid with points labelled by x and y coordinates. The “origin” (0,0) of this grid is at the top left corner. The x coordinates increase as you move from left to right, and the y coordinates increase as you move from top to bottom. The 2D drawing context provides commands for drawing on this grid. Try adding this command:

```
context.fillRect(20, 50, 200, 100);
```

This creates a filled Rectangle of width 200 pixels and height 100 pixels with its top left corner at coordinate (20, 50). The default drawing colour is black.

To change the drawing colour to blue, use this command. You will need to add it before the `fillRect` command.

```
      context.fillStyle = "blue";
```

The lines below write Hello World on the canvas in white filled text, Arial font, 30 pixels high, with the bottom left of the text positioned at (40, 110).

```
      context.fillStyle = "white";
      context.font = "30px Arial";
      context.fillText("Hello World", 40, 110);
```

At this point, your document should contain a script which draws a blue box on the canvas containing the words Hello World in white text.

For the next step you are going to animate your drawing. The animation will make the blue box glide across the canvas in a straight line, reversing direction whenever it bounces off an edge. This will test your JavaScript skills. The code below is given as a starting point:

Comment out the commands you wrote for drawing the blue box with text. (Don't delete them as you will need to refer to them later.) Add the script below to the body of your document:

```
<script>
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
var x = 20;    // x coordinate of box position
var y = 50;    // y coordinate of box position
var dx = 2;    // amount to move box to the right
var dy = 4;    // amount to move box down
var WIDTH = 500;     // width of canvas
var HEIGHT = 200;    // height of canvas
var MESSAGE_WIDTH = 200;    // width of message
var MESSAGE_HEIGHT = 100;  // height of message
animate();     // run the animation
</script>
```

This script begins by declaring the usual variables to hold the canvas and the context, and then adds a number of variables and constants to control the animation. Hopefully, the comments are self explanatory.

The last line is a call to a function called `animate` that you have not written yet. Let's add that next. Add the script below to the *head* of your document. (Note, this script *could* be added to the body but we are following the usual practice of putting functions into the head rather than the body.)

The `animate` function is given in full. It uses a library function `setInterval` to repeatedly call the `drawMessage` function every 10 milliseconds. This is how the animation effect is achieved. The `drawMessage` function draws each animation "frame" and is left for you to complete.

```
<script>

function drawMessage()
{
   // Complete this function yourself. It needs to:

   // 1) Clear the canvas by drawing a white rectangle that
   // fills it entirely,

   // 2) Draw the message (blue rectangle enclosing
   // white text) at position (x, y)

   // 3) Update x and y for the next time. This needs
   // a bit of logic. If adding dx to x would cause the message
   // to move off the canvas (either too far right or left),
   // then dx should be set to -dx. Similarly, if adding dy to
   // y would make the message move off the canvas, dy should
   // be set to -dy. After making these checks, add dx to x
   // and add dy to y.
}


function animate()
{
  return setInterval(drawMessage, 10);
}

</script>
```

This has been a very basic introduction to animation in HTML5 with JavaScript. For serious animation, we recommend that you explore JavaScript libraries such as JQuery rather than programming everything from scratch. However, these are outside the scope of CSCU9A2.

CHECKPOINT [HTML5]

Demonstrate (both in a browser and in TextPad) your document animations.html, the illusion video with a button to toggle its size and a canvas containing an animated message.