CSS3 Transformations and Animations

Overview

In this lab, you'll implement two web pages that allow the user to apply transformations when viewing an image:

- The first web page, `SimpleViewer.html`, will allow the user to click buttons to rotate, translate, and scale the image.
- The second web page, `InteractiveViewer.html`, will allow the user to use mouse and keyboard gestures to achieve the same effect. The net result will be similar to the way you can view maps in websites such as Google Maps and Bing Maps.

Roadmap

There are 5 exercises in this lab, of which the last two exercises are "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Simple transformations
2. Setting the transformation origin
3. Defining a transition for a transformation
4. Interactive transformations (if time permits)
5. Additional suggestions (if time permits)

Familiarization

Open a browser window and browse to `SimpleViewer.html` in the *Solutions* folder. The web page displays an image enclosed inside a `<div>` element. The image is actually bigger than the `<div>`, and it has negative `margin-left` and `margin-top` CSS properties relative to the `<div>` element. We've defined some CSS properties to ensure the image is clipped to the `<div>` element when it is displayed on the web page.

The web page has several buttons that allow the user to transform the image. Click each of the buttons to see the effects. Note the following points:

- Each button applies a single transformation to the image (i.e. the transformations are not cumulative). For example, if you click *Turn clockwise* twice, the second click seems to have no effect because the image has already been rotated.
- The transformation origin is the top-left corner of the image.
- There's a transition of 3 seconds for each transformation, for a smooth user experience.

Now switch to the *Start* folder and open `SimpleViewer.html` in the text editor. The `<body>` of the web page contains the following elements:

- A series of buttons, each of which applies a specific transformation. Each button has a `data-description` attribute that describes the purpose of the button, and a `data-class-to-add` attribute that specifies the CSS classes to apply to the image when that button is clicked.
- A `<div>` element named `myImageContainer`. The `<div>` element contains the image to be transformed.

Now go to the top of the web page and locate the `<style>` element. Note these CSS rules:

- The `#myImageContainer` rule defines CSS properties for the `<div>` element. Note the `overflow: hidden` CSS property, which ensures any nested elements are hidden if they overflow the confines of the `<div>` element. This is how we ensure the image is clipped to the viewing region of the `<div>` element.
- The `#myImageContainer img` rule defines CSS properties for the `<img>` element. Note the `width` and `height` are much larger than the `<div>` element, and the `margin-left` and `margin-top` are negative.
- There are a series of empty CSS rules corresponding to the buttons on the web page. You'll implement transformations in these CSS rules shortly.

Now skim down and locate the `onload()` function in the `<script>` element. The function performs the following tasks:

- Sets the label for each button, based on its `data-description` attribute.
- Installs a `click` handler function for each button. The handler function gets the `data-class-to-add` attribute for the clicked button, and applies that CSS class to the image.

Exercise 1: Simple transformations

Implement transformations in the CSS rules as indicated by the TODO comments:

TODO	CSS rule	Transformation to perform
`TODO 1a`	`.rotateClockwise`	Clockwise rotation of 15 degrees
`TODO 1b`	`.rotateAntiClockwise`	Anti-clockwise rotation of 15 degrees
`TODO 1c`	`.moveLeft`	Translation to the left of 10%
`TODO 1d`	`.moveRight`	Translation to the right of 10%
`TODO 1e`	`.zoomIn`	Scaling of 1.5, to zoom in
`TODO 1f`	`.zoomOut`	Scaling of 0.6, to zoom out

Make sure you also define vendor-specific transformations too (e.g. `-ms-transform`, `-webkit-transform`, etc.). Then open the *Start* version of `SimpleViewer.html` in the browser. Verify the image is transformed when you click each button. The transformations will be based on the centre of the image at the moment, and there is no smooth transition yet.

Exercise 2: Setting the transformation origin

Locate the `.topLeftOrigin` CSS rule. In this rule, set the `transform-origin` CSS property to the top-left corner of the target element. Make sure you also set the vendor-specific versions of `transform-origin` too.

In the browser, refresh `SimpleViewer.html`. Verify all the transformations are based on the top-left corner of the image (which is located outside the confines of the `<div>`).

Exercise 3: Defining a transition for a transformation

Locate the `#myImageContainer img` rule and define a `transition` CSS property:

- Transition target property: `transform`
- Transition duration: `3s` (i.e. 3 seconds)
- Transition timing effect: `ease-in-out`

Make sure you also set the vendor-specific versions of `transition` too. For example, set `-ms-transition` so that it targets the `-ms-transform` property, etc.

Refresh `SimpleViewer.html` in the browser. Verify all the transformations are now applied smoothly over 3 seconds.

Exercise 4 (If time permits): Interactive transformations

In the browser, browse to `InteractiveViewer.html` in the *Solutions* folder. The web page is similar to the previous one, except that the user can now interact directly with the image to perform transformations. Try the following user interactions:

- Move the mouse on the image, whilst holding down either the mouse button or the SHIFT key. This causes the image to scroll left, right, up, or down, based on the mouse movement. This provides a fast and intuitive mechanism to allow the user to move the image around.
- Press the > or < keys to rotate the image clockwise or anti-clockwise.
- Press the + or - keys on the number keypad to zoom-in or zoom-out.
- Press the SPACE key to cancel all transformations.

Note that these transformations are cumulative. For example, every time you press the > key, it increases the angle of rotation.

Now switch to the *Start* folder and open `InteractiveViewer.html` in the text editor. Locate the `<script>` element and note the following points:

- The `currentTransform` global object has properties named `angle`, `scale`, `xOffset`, and `yOffset` to keep track of the transformation state for the image.
- The `onload()` function handles various keyboard and mouse events:
    - For keystrokes, the function updates the appropriate property on the `currentTransform` object, and then calls `applyTransformation()` to apply the transformation according to the current transformation state.
    - For mouse movements, if the mouse button or SHIFT key is currently depressed, the function calculates how much the mouse has moved since the last `mousemove` event, and then updates the `xOffset` and `yOffset` properties to translate the image by that amount. The function then calls `applyTransformation()` to realize this translation.
    - Note: There's some interesting code that handles SHIFT-key and mouse-button events. Take a good look at the code if you're interested ☺.

Now locate the `applyTransformation()` function. Complete the implementation of this function, as described by the TODO comments, so that it applies a scaling, rotation, and translation transformation on the image, as specified by the current state of the `currentTransform` object.

Open the web page in the browser and verify all the keystrokes and mouse interactions work.

Exercise 5 (If time permits): Additional suggestions

Implement a function named `getTransformProperty()`, as indicated by the TODO comments. The purpose of this function is to return the name of the transform property to use in the current browser (i.e. `'transform'`, `'msTransform'`, etc.). Refactor the other code in the web page to use `getTransformProperty()` as appropriate.

Refresh the web page in the browser and verify it all still works correctly. You might also like to try it out in various other browsers to ensure the web page is portable ☺.