Translation

Allows movement of an element in a up, down, left, right direction relative to its current position, as opposed to relative to its parent.

transform: translate(45px, -45px);

This moves the element in the x direction (right) 45px and down 45px. A single direction can be specified by: transform: translateX(45px);

Transform doesn’t work on inline elements unless display: inline-block is specified

Scaling

Scale(x,y) shrinks or grows an element

Rotation

transform: rotate(10deg)

nSkew

transform: skew(15deg, 4deg);

Transform origin

Default is set to transform-origin: 50% 50%, syntax is the same as background-position

The order of transforms is important, translate then rotate is different from rotate then translate

Transitions

1. Declare the original state of the element in the default style declaration.
2. Declare the final state of your transitioned element; for example, a :hover state.
3. Include the transition functions in your default style declaration using the transition properties, including: transition-property , transition-duration , transition-timing-function , and transition-delay .

.ad-ad2 h1 span {

-webkit-transition-property: -webkit-transform;

transition-property: transform;

}

Transition-delay

A ‘timer’ can be set to wait before a transition begins. A negative value can be supplied causing the animation to start immediately, but part way through the animation.

Animations

@keyframes myAnimation {

/\* put animation keyframes here \*/

}

Canvas, SVG, Drag and Drop

Canvases are used for drawing images on

<canvas id="myCanvas" class="myCanvas" width="200" height="200">

Sorry! Your browser doesn’t support Canvas.

</canvas>

var canvas = document.getElementById("myCanvas");

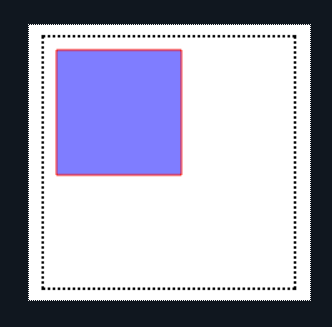
var context = canvas.getContext("2d");

context.strokeStyle = "red";

context.fillStyle = "rgba(0, 0, 255, 0.5)";

context.fillRect(10, 10, 100, 100);

context.strokeRect(10, 10, 100, 100);



Complex Shapes

These use “path” as there is no explicit name for other shapes. The three steps for defining these shapes are:

1. Layout the path (create the shape)
2. Stroke the path (make it visible)
3. Fill the path

Creating a circle with an arc

function drawCircle(canvas) {

var canvas = document.getElementById("myCanvas");

var context = canvas.getContext("2d");

context.beginPath();

context.arc(50, 50, 30, 0, Math.PI\*2, true);

}

The signature for the arc method is: arc(x, y, radius, startAngle, endAngle, anticlockwise) . Anticlockwise is optional

Context.closePath() end the path

Images can be used in canvases

var canvas = document.getElementById("myCanvas");

var context = canvas.getContext("2d");

var image = document.getElementById("myImageElem");

// draw the image at x=0 and y=0 on the canvas

context.drawImage(image, 68, 68);

var imageData = context.getImageData(0, 0, 1, 1);

var pixelData = imageData.data;

console.log(pixelData.length);

Drag and Drop

Making an element draggable is as simple as setting the draggable attribute to true

<img data-src="https://learnable-static.s3.amazonaws.com/premium/reeedr/books/html5-css3-for-the-real-world-2nd-edition/images/computer-mouse-pic.svg" width="30"

↵alt="mouse treat" id="mouse3" draggable="true">

To accept a dropped element two event listeners are to be added, dragover and drop.

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

cat.addEventListener("dragover", function(event) {

event.preventDefault();

});

cat.addEventListener("drop", function(event) {

var mouseHash = {

mouse1: 'NOMNOMNOM',

mouse2: 'Meow',

mouse3: 'Purrrrrr ...'

};

}