CANVAS

CANVAS IS A COMPLEX ELEMENT THAT BASICALLY CONSISTS OF 2 PARTS:
HTML <canvas id="cnv" width="600" height="300"></canvas>
JS
var c = document.getElementById("cnv");
var ctx = c.getContext("2d");

REQUIRED ATTRIBUTES FOR <CANVAS>
id
Width
Height

TYPES OF CONTENT:
Fallback content
<canvas id="cnv" width="600" height="300">Your browser doesn’t support canvas!</canvas>
2D content
var ctx = canvas.getContext('2d');
3D content
var ctx = canvas.getContext('webgl');

SHAPES

RECTANGLE
To draw a rectangle:
rect(x, y, width, height)
fillRect(x, y, width, height)
strokeRect(x, y, width, height)
clearRect(x, y, width, height)

PATH
To draw a path:
beginPath()
Use Path methods
closePath()
stroke()/fill()
Path methods
moveTo()
lineTo()
bezierCurveTo()
quadraticCurveTo()
arc()
arcto()
elipse()
rect()

STYLES & COLORS

COLORS:
fillStyle = color
strokeStyle = color
Transparency:
globalAlpha = transparencyValue
Line styles:
lineWidth = value
lineCap = type

TEXT

DRAWING TEXT:
fillText(text, x, y [, maxWidth])
strokeText(text, x, y [, maxWidth])

STYLING TEXT:
font = value
textAlign = value
textBaseline = value
direction = value

ADVANCED TEXT MEASUREMENT
measureText()

IMAGES

CANVAS API CAN USE ANY OF THE FOLLOWING DATA TYPES
HTMLImageElement
HTMLVideoElement
HTMLCanvasElement

GET AN IMAGE:
from the same page:
from other domain:
use another canvas element:
Create images from scratch
Embedding an image via data: url.
Using frames from scratch

DRAW AN IMAGE:
drawImage(image, x, y)

SCALE AN IMAGE:
drawImage(image, x, y, width, height)

SLICE AN IMAGE:
drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight)
CONTROL IMAGE SCALING BEHAVIOR
ctx.mozImageSmoothingEnabled = false;
ctx.webkitImageSmoothingEnabled = false;
ctx.msImageSmoothingEnabled = false;
ctx.imageSmoothingEnabled = false;

TRANSFORMATIONS
save()
restore()
translate(x, y)
rotate(angle)
scale(x, y)
transform(a, b, c, d, e, f)
setTransform(a, b, c, d, e, f)
resetTransform()

COMPOSITING AND CLIPPING

source-over
source-in
source-out

source-atop
destination-over
destination-in

copy
xor
destination-out

destination-atop
lighter

ANIMATION

BASIC ANIMATION STEPS:
Clear the canvas
Save the canvas state
Draw animated shapes
Restore the canvas state

SCHEDULE UPDATES:
setInterval(function, delay)
setTimeout(function, delay)
requestAnimationFrame(callback)

PIXEL MANIPULATION

createImageData()
getImageData()
pullImageData()
drawImage()

Saving images
Creates a PNG image canvas.toDataURL(‘image/png’)

HIT REGIONS AND ACCESSIBILITY

Hit regions (experimental)
CanvasRenderingContext2D.addHitRegion()
CanvasRenderingContext2D.removeHitRegion()
CanvasRenderingContext2D.clearHitRegions()

Focus ring (experimental):
CanvasRenderingContext2D.drawFocusIfNeeded()
CanvasRenderingContext2D.scrollPathIntoView()

USEFUL TIPS

Pre-render similar primitives or repeating objects on an off-screen canvas
Avoid floating-point coordinates and use integers instead
Don’t scale images in drawImage
Use multiple layered canvases for complex scenes
CSS for large background images
Scaling canvas using CSS transforms
Use the moz-opaque attribute (Gecko only)
Batch canvas calls together
Avoid unnecessary canvas state changes.
Render screen differences only, not the whole new state.
Avoid the shadowBlur property whenever possible.
Avoid text rendering whenever possible.
Try different ways to clear the canvas (clearRect() vs.
fillRect() vs. resizing the canvas)
With animations, use window.requestAnimationFrame() instead of window.setTimeout().
Be careful with heavy physics libraries

Source:
http://www.w3schools.com/tags/tag_canvas.asp
http://www.html5canvastutorials.com/
https://www.sitedepoint.com/html5-canvas-tutorial-introduction/
https://en.wikipedia.org/wiki/Canvas_element