CANVAS CONSISTS OF 2 PARTS

HTML

```
<canvas id="cnvs" width="300" height="300"></canvas>
```

JS

```
var c = document.getElementById(<cnv>);
var ctx = c.getContext("");
```

REQUIRED ATTRIBUTES

ID: Although not required, supplying an ID is recommended as it eases the process of identifying it within a script

WIDTH: Initial width is 300px

HEIGHT: Initial height is 300px

TYPES OF CONTENT

Fallback content

```
<canvas id="cnvs" width="300" height="300">Your browser doesn't support canvas!</canvas>
```

2D content

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

3D content

```
var ctx = canvas.getContext('webgl');
```

SHAPES

Canvas supports 2 primitive shapes: rectangles and paths

RECTANGLE

4 WAYS TO CREATE A RECTANGLE

Adds a rectangular path to a currently open path rect(x, y, width, height)

Filled rectangle fillRect(x, y, width, height)

A rectangular outline strokeRect(x, y, width, height)

Makes the rectangular area transparent clearRect(x, y, width, height)

WHERE:

X The x axis of the coordinate

Y The y axis of the coordinate

WIDTH The rectangle's width

HEIGHT The rectangle's height

PATH

A path is a list of points connected by segments of lines that can be of varying shapes, curves, widths, and colors. Both paths and subpaths can also be closed.

TO MAKE SHAPES USING PATHS:

Create the path beginPath()

Use path methods to draw into the path

Use stroke() or fill() the path to render it

Close the path closePath()

PATH METHODS:

moveTo() lineTo() bezierCurveTo() arc()

quadraticCurveTo() arcTo() ellipse() rect()

For series of paths use Path2D object!

Or use SVG path data for both SVG and canvas.

STYLES & COLOR

COLORS

Sets the styles used when filling shapes fillStyle = color

Sets the styles for shape outline strokeStyle = color

TRANSPARENCY

Applies the specified transparency value to all future shapes drawn on the canvas globalAlpha = transparencyValue

PATTERNS

Creates and returns a new canvas pattern object createPattern(image, type)

CANVAS FILL RULES

Nonzero-rule Even-odd rule

LINE STYLES

lineWidth = value

miterLimit = value

getLineDash()

setLineDash(segments)

lineDashOffset = value

butt round square

lineCap = type

round bevel miter

lineJoin = type

SHADOWS

shadowOffsetX = float

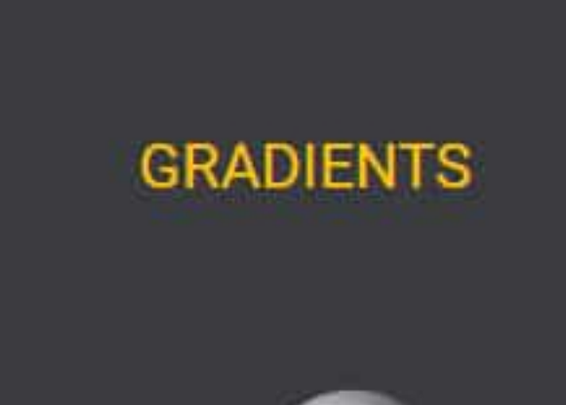
shadowOffsetY = float

shadowBlur = float

shadowColor = color

GRADIENTS

Creates a linear gradient object createLinearGradient(x1, y1, x2, y2)



Creates a radial gradient createRadialGradient(x1, y1, r1, x2, y2, r2)

Creates a new color stop gradient.addColorStop(position, color)

TEXT

DRAWING TEXT

CODING-DUDE.COM

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 DATA TYPES

HTMLImageElement

HTMLVideoElement

HTMLCanvasElement

CONTROL IMAGE SCALING BEHAVIOR

```
ctx.mozImageSmoothingEnabled = false;
ctx.webkitImageSmoothingEnabled = false;
ctx.msImageSmoothingEnabled = false;
ctx.imageSmoothingEnabled = false;
```

GET AN IMAGE

FROM THE SAME PAGE:

the document.images collection

the document.getElementsByTagName() method

the ID of the image document.getElementById()

FROM OTHER DOMAIN:

Does the hosting domain permits cross-domain access to the image?

YES: use crossorigin attribute of an <img>

NO: using the image can taint the canvas

BY USING ANOTHER CANVAS ELEMENT:

document.getElementsByTagName()

document.getElementById()

CREATE FROM SCRATCH

Create new HTMLImageElement objects in our script

EMBEDDING AN IMAGE VIA DATA: URL

USING FRAMES FROM A <VIDEO></VIDEO>

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)

Given an image, this function first takes the area of the source image specified by the rectangle with a top-left corner of *(sx,sy)* combined with a width and height of *sWidth* and *sHeight* to draw it into the canvas. Then, the function places it on a canvas at *(dx,dy)* and scales it to the size specified by *dWidth* and *dHeight*.

TRANSFORMATIONS

save() Saves the entire state of the canvas

restore() Restores the most recently saved canvas state

translate(x, y) Moves the canvas and its origin on the grid

rotate(angle) Rotates the canvas clockwise around the current origin by the angle number of radians

scale(x, y) Scales the canvas units by x horizontally and by y vertically

transform(a, b, c, d, e, f) Multiplies the current transformation matrix with the matrix described by its arguments

setTransform(a, b, c, d, e, f) Resets the current transform to the identity matrix, and then invokes the transform() method with the same arguments

resetTransform() Resets the current transform to the identity matrix

COMPOSITING & CLIPPING

globalCompositeOperation = type Sets the type of compositing operation to apply when drawing new shapes. 'Type' is used to identify which of the twelve compositing operations to use.

clip() Turns the path currently being built into the current clipping path

source-over source-in source-out source-atop destination-over destination-in

destination-out destination-atop lighter copy xor

ANIMATION

BASIC ANIMATION STEPS

1. Clear the canvas
2. Save the canvas state
3. Draw animated shapes
4. Restore the canvas state

SCHEDULE UPDATES

setInterval(function, delay) Repeatedly executes the function specified in every delay milliseconds

setTimeout(function, delay) Executes the function specified by function in delay milliseconds

requestAnimationFrame(callback) Informs the browser to perform an animation and requests the browser call a specified function to update an animation before the next repaint

PIXEL MANIPULATION

CREATE IMAGE DATA OBJECT

createImageData()

GET THE PIXEL DATA FROM THE CONTENT

getImageData()

PAINTING PIXEL DATA INTO CONTENT

putImageData()

ZOOMING AND ANTI-ALIASING

drawImage()

SAVING IMAGES

- Default setting. Creates a PNG image. canvas.toDataURL('image/png')
- Creates a JPG image canvas.toDataURL('image/jpeg', quality)
- Creates a Blob object representing the image contained in the canvas canvas.toBlob(callback, type, encoderOptions)

HIT REGIONS & ACCESSIBILITY

HIT REGIONS

Adds a hit region to the canvas ctx.addHitRegion()

Removes the hit region with the specified id from the canvas ctx.removeHitRegion()

Removes all hit regions from the canvas ctx.clearHitRegions()

FOCUS RING

If a given element is focused, this method draws a focus ring around the current path ctx.drawFocusIfNeeded()

Scrolls the current path or given path into the view ctx.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.setInterval()

Be careful with heavy physics libraries

CANVAS INSPIRATION








SOURCES

https://developer.mozilla.org/en/docs/Web/API/Canvas_API/Tutorial

http://www.w3schools.com/tags/tag_canvas.asp

http://www.html5canvastutorials.com/

https://www.sitepoint.com/html5-canvas-tutorial-introduction/

https://www.tutorialspoint.com/html5/html5_canvas.htm

http://sixrevisions.com/html/canvas-element/

https://en.wikipedia.org/wiki/Canvas_element