Canvas

How to use HTML5 canvas 2D API

Canvas for fun

- Creative coding
    - awesome-canvas
    - Matt DesLauriers
    - Sketch.js
    - PointPatterns
    - Creative coding basics
- Audio Visualization
    - A curated list about Audio Visualization
    - Build a Music Visualizer with the Web Audio API
    - How To: Music Visualizer (Web Audio API)

Get Ready

- Create canvas element
    - <canvas id="canvas" width="500" height=“300”></canvas>
    - var canvas = document.createElement(‘canvas’);
    - default width 300, height 150
- canvas.getContext(‘2d’): CanvasRenderingContext2D
- CanvasRenderingContext2D
- Ready for high DPI
    - var ratio = window.devicePixelRatio
    - ctx.backingStorePixelRatio
    - canvas.width/height = width/height * ratio
    - canvas.style.width/height = width/height + ‘px';
    - ctx.scale(ratio, ratio);

Clean

- clearRect(x, y, w, h)
  - clear canvas
- canvas.width/height = canvas.width/height
  - clear canvas and state
  - need scale again for Hight DPI
- dat.gui

Path

- Path
    - A path has a list of zero or more subpaths
- Current path
    - not part of the drawing state
    - can only be reset using the beginPath()
- Subpath
    - consists of a list of one or more points, connected by straight or curved lines
    - and a flag indicating whether the subpath is closed or not
    - A closed subpath is one where the last point of the subpath is connected to the first point of the subpath by a straight line
- beginPath()
- closePath()
    - mark the last subpath as closed, create a new subpath whose first point is the same as the previous subpath's first point, and finally add this new subpath to the path

Path

- moveTo(x, y)
    - create a new subpath with the specified point as its first (and only) point
- lineTo(x, y)
- quadraticCurveTo(cpx, cpy, x, y)
- bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)
- arcTo(x1, y1, x2, y2, radius)
    - Create an arc between two tangents { from: {x: x0, y: y0}, to: {x: x1, y: y1} }, { from: {x: x1, y: y1}, to: {x: x2, y: y2} }
- arc(x, y, radius, startAngle, endAngle, anticlockwise)
    - If the context has any subpaths, then the method must add a straight line from the last point in the subpath to the start point of the arc
- ? ellipse(x, y, radiusX, radiusY, rotation, startAngle, endAngle, anticlockwise)
- rect(x, y, w, h)
    - create a new subpath containing just the four connected points
    - mark the subpath as closed
    - create a new subpath with the point (*x*, *y*) as the only point in the subpath
- roundedRect

Path2D

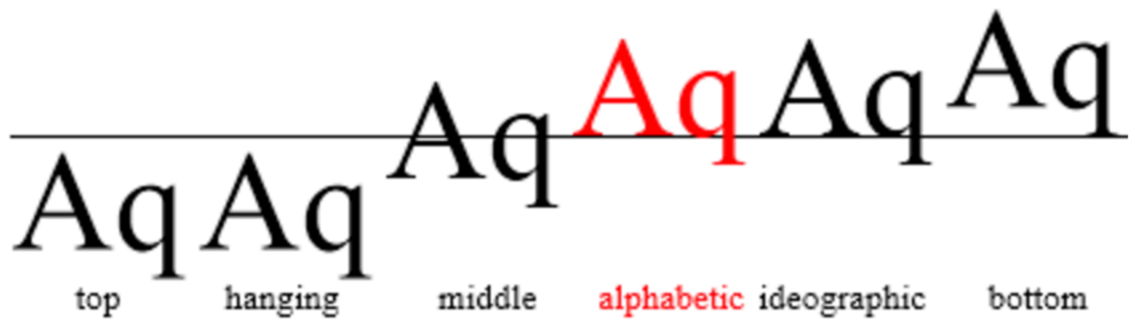
- HTML 5 Canvas Polyfill
- Can I use Path2D
- API
    - ctx.fill([fillRule]); ctx.fill(path [, fillRule]);
    - ctx.stroke(); ctx.stroke(path);
    - ctx.isPointInPath(x, y [, fillRule]); ctx.isPointInPath(path, x, y [, fillRule]);
    - ctx.isPointInStroke(x, y); ctx.isPointInStroke(path, x, y);
    - ctx.clip(path [, fillRule]); ctx.clip([fillRule]);

Draw Path

- strokeRect(x, y, w, h), fillRect(x, y, w, h)
    - without affecting the current default path
- fill(? fillRule)
    - nonzero: non-zero winding rule, default
    - evenodd: even-odd winding rule
- stroke()
- clip()
    - Create a new clipping region by calculating the intersection of the current clipping region and the area described by the path, using the non-zero winding number rule
    - Open subpaths must be implicitly closed when computing the clipping region, without affecting the actual subpaths
    - The new clipping region replaces the current clipping region

Text

- font
- textAlign: default start
  - start, end, left, right, center
- textBaseline: default alphabetic
  - top, hanging, middle, alphabetic, ideographic, bottom
- fillText(text, x, y, maxWidth)
- strokeText(text, x, y, maxWidth)
- measureText(text): TextMetrics
  - width
  - height



State

- save()
    - transformation matrix
    - clipping region
    - strokeStyle, fillStyle
    - globalAlpha, globalCompositeOperation
    - lineWidth, lineCap, lineJoin, miterLimit, lineDash
    - shadowOffsetX, shadowOffsetY, shadowBlur, shadowColor
    - font, textAlign, textBaseline
- restore()

Transformations

- scale(x, y)
- rotate(angle)
- translate(x, y)
- transform(a, b, c, d, e, f)
- setTransform(a, b, c, d, e, f)

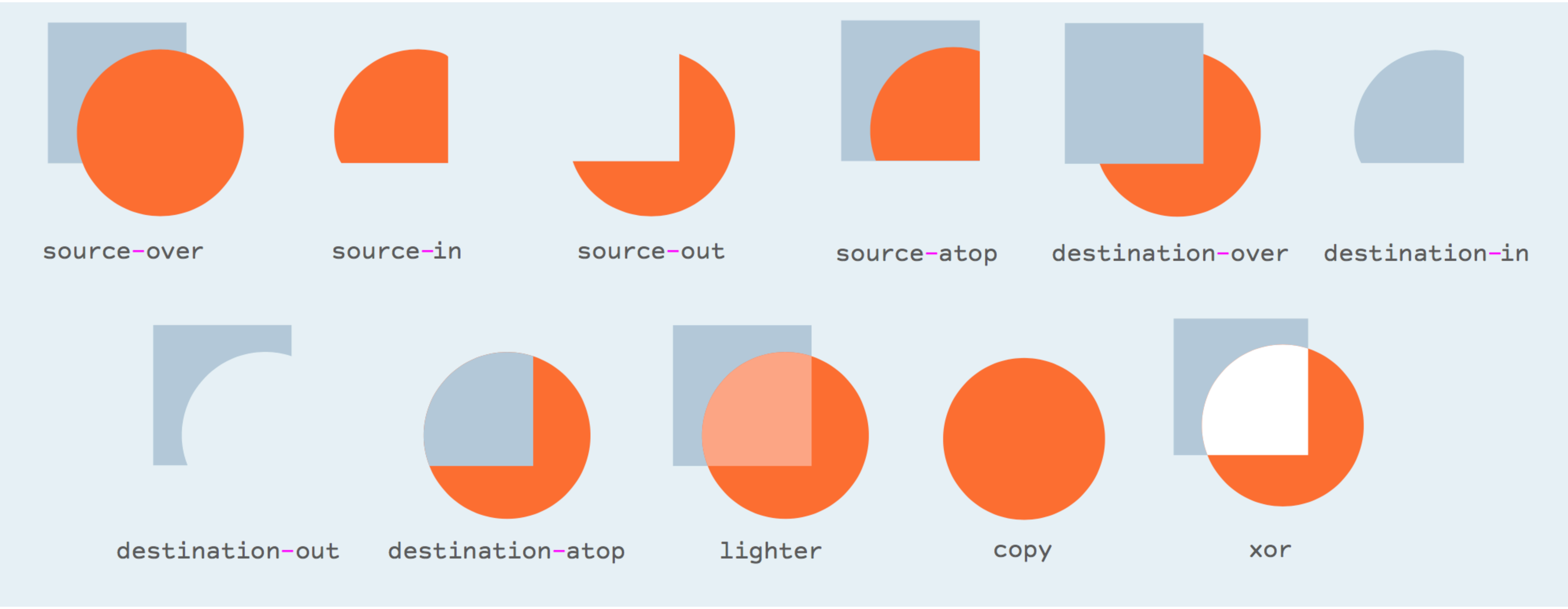
Composting

- globalAlpha: default 1.0
- globalCompositeOperation: default 'source-over'

globalCompositeOperation

- source-over: The default. New content is drawn over existing content.
- source-in: New content is only drawn where existing content was non-transparent.
- source-out: New content is drawn only where there was transparency.
- source-atop: New content is drawn only where its overlap existing content.
- destination-over: Opposite of source-over. It acts as if new content is drawn "behind" existing content.
- destination-in: Opposite of source-in. Existing content is drawn only where new content is non-transparent.
- destination-out: Opposite of source-out. Existing content is drawn only where new content is transparent. Acts as if existing content is drawn everywhere except the where the new content is.
- destination-atop: Opposite of source-atop. New content is drawn, and then old content is drawn only where it overlaps with new content.
- lighter: Where new content overlaps old content, color is determined by adding the color values.
- copy: New content replaces all old content.
- xor: New content is drawn where old content is transparent. Where the content of both old and new are not transparent, transparency is drawn instead.

globalCompositeOperation



Colors and Styles

- strokeStyle: default '#000000'
- fillStyle: default '#000000'
- Value can be:
    - 'red', '#ff0000', 'rgb(255,0,0)', 'rgba(255,0,0,1)'
    - CanvasGradient
    - CanvasPattern

Gradient

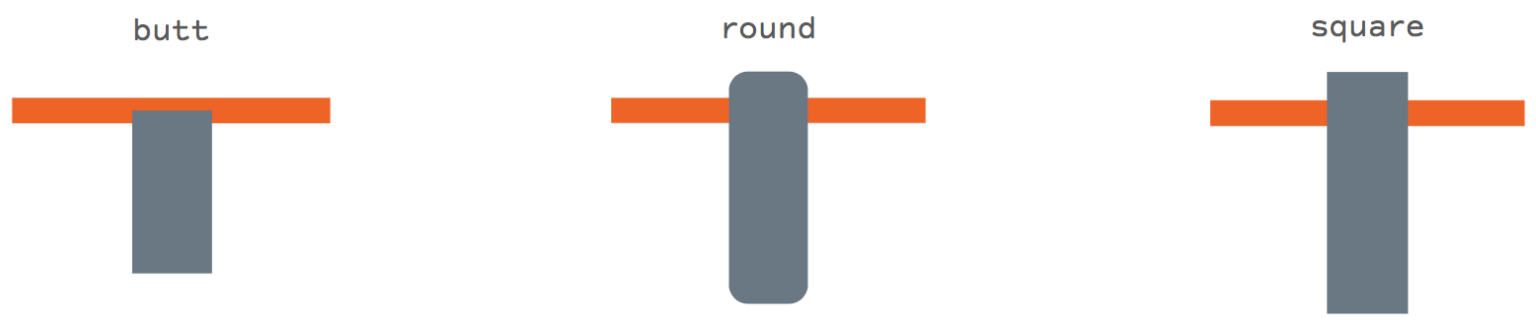
- createLinearGradient(x0, y0, x1, y1)
- createRadialGradient(x0, y0, r0, x1, y1, r1)
- CanvasGradient
    - addColorStop(offset, color)
- HTML5 Canvas Gradient Creator

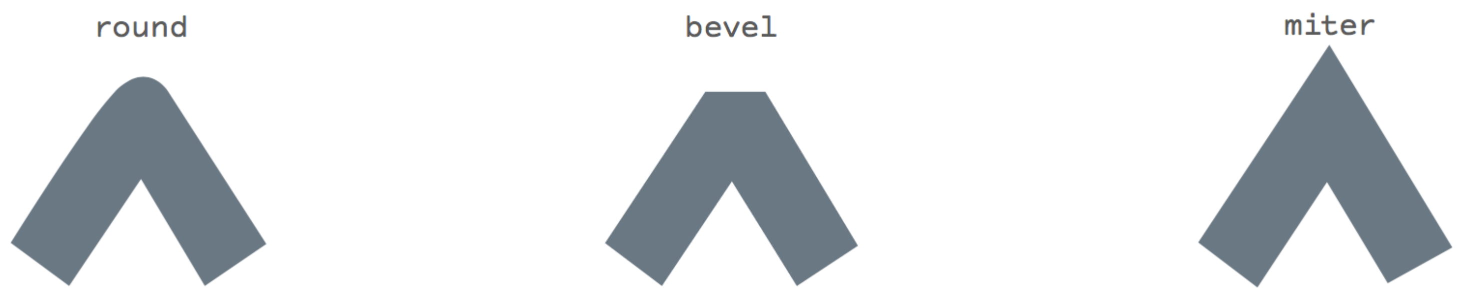
Pattern

- createPattern(image, repetition): CanvasPattern
    - image:
        - HTMLImageElement
        - HTMLCanvasElement
        - HTMLVideoElement
        - ImageBitmap
    - repetition
        - repeat: default
        - repeat-x
        - repeat-y
        - no-repeat

Line Styles

- lineWidth: default 1
- lineCap: default 'butt'
    - butt
    - round
    - square
- lineJoin: default 'miter'
    - round
    - bevel
    - miter
- miterLimit: default 10
- setLineDash(segments): IE11
- lineDashOffset: 0, IE11





Shadows

- shadowColor: default 'rgba(0, 0, 0, 0)'
- shadowOffsetX: default 0
- shadowOffsetY: default 0
- shadowBlur: default 0

Images

- drawImage(image, dx, dy)
- drawImage(image, dx, dy, dw, dh)
- drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh)
- invoke drawImage after image loaded

Pixel manipulation

- createImageData(sw, sh): ImageData
- createImageData(imagedata)
- getImageData(sx, sy, sw, sh)
- putImageData(imagedata, dx, dy)
- ? putImageData(imagedata, dx, dy, dirtyX, dirtyY, dirtyWidth, dirtyHeight)
- ImageData
  - width, height, data

Export Image

- canvas.toDataURL(‘image/png’)
  - data:image/png;base64,
  - window.open
  - img.src
- ? canvas.toBlob(callback, type)

Unit test Canvas

- jsdom
- node-canvas
- configuration
- jest-environment-jsdom-fourteen

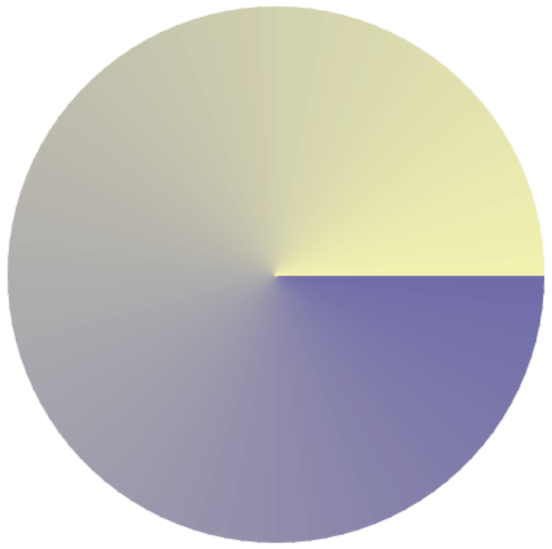
Tips

- offset 0.5 pixel when lineWidth is even
- beginPath first before draw any path
- save first before change state
- usage of putImageData
- usage of globalAlpha and globalCompositeOperation
- how to hitTest

Performance

- stroke once if same style / set style once
- drawImage is faster than text or path
- cache text or shape to canvas if no need change or scale
- draw images on integer coordinates
    - ctx.drawImage(yourImage, x | 0, y | 0);
- prefer drawImage to putImageData
- prefer requestAnimationFrame to setTimeout
- only redraw changed and visible on screen
- use background, middle, foreground canvas

Homework



- gradient from #706caa to #f2f2b0