

HTML5 CANVAS CHEAT SHEET

This cheat sheet summarizes the complete HTML5 Canvas API for the 2D context, based on to the [W3C HTML5 Canvas Spec](#). It also provides techniques for handling common procedures.

- **HTML5 CANVAS ELEMENT**

Html5 canvas element

```
<canvas id="myCanvas" width="500" height="300">
```

Html5 canvas element with fallback content

```
<canvas id="myCanvas" width="500" height="300">
  your browser doesn't support canvas!
</canvas>
```

2d context

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

Webgl context (3d)

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

- **SHAPES**

Draw rectangle

```
context.rect(x, y, width, height);
context.fill();
context.stroke();
```

Fill rectangle shorthand

```
context.fillRect(x, y, width, height);
```

Stroke rectangle shorthand

```
context.strokeRect(x, y, width, height);
```

Draw circle

```
context.arc(x, y, radius, 0, Math.PI * 2);
context.fill();
context.stroke();
```

- **STYLES**

Fill

```
context.fillStyle = 'red';
context.fill();
```

Stroke

```
context.strokeStyle = 'red';
context.stroke();
```

Linear gradient

```
var grd = context.createLinearGradient(x1, y1, x2, y2);
grd.addColorStop(0, 'red');
grd.addColorStop(1, 'blue');
context.fillStyle = grd;
context.fill();
```

Radial gradient

```
var grd = context.createRadialGradient(x1, y1, radius1, x2, y2, radius2);
grd.addColorStop(0, 'red');
grd.addColorStop(1, 'blue');
context.fillStyle = grd;
context.fill();
```

Pattern

```
var imageObj = new Image();
imageObj.onload = function() {
    var pattern = context.createPattern(imageObj, 'repeat');
    context.fillStyle = pattern;
    context.fill();
};
imageObj.src = 'path/to/my/image.jpg';
```

Line Join

```
context.lineJoin = 'miter|round|bevel';
```

Line Cap

```
context.lineCap = 'butt|round|square';
```

Shadow

```
context.shadowColor = 'black';
context.shadowBlur = 20;
context.shadowOffsetX = 10;
context.shadowOffsetY = 10;
```

Alpha (Opacity)

```
context.globalAlpha = 0.5; // between 0 and 1
```

• COLOR FORMATS

String

```
context.fillStyle = 'red';
```

Hex Long

```
context.fillStyle = '#ff0000';
```

Hex Short

```
context.fillStyle = '#f00';
```

RGB

```
context.fillStyle = 'rgb(255,0,0)';
```

RGBA

```
context.fillStyle = 'rgba(255,0,0,1)';
```

• PATHS

Begin Path

```
context.beginPath();
```

Line

```
context.lineTo(x, y);
```

Arc

```
context.arc(x, y, radius, startAngle, endAngle, counterClockwise);
```

Quadratic curve

```
context.quadraticCurveTo(cx, cy, x, y);
```

Bezier curve

```
context.bezierCurveTo(cx1, cy1, cx2, cy2, x, y);
```

Close Path

```
context.closePath();
```

• IMAGES

Draw Image with default size

```
var imageObj = new Image();
imageObj.onload = function() {
    context.drawImage(imageObj, x, y);
};
imageObj.src = 'path/to/my/image.jpg';
```

Draw image and set size

```
var imageObj = new Image();
imageObj.onload = function() {
    context.drawImage(imageObj, x, y, width, height);
};
```

```
imageObj.src = 'path/to/my/image.jpg';
```

Crop image

```
var imageObj = new Image();
imageObj.onload = function() {
    context.drawImage(imageObj, sx, sy, sw, sh, dx, dy, dw, dh);
};
imageObj.src = 'path/to/my/image.jpg';
```

- **TEXT**

Fill Text

```
context.font = '40px Arial';
context.fillStyle = 'red';
context.fillText('Hello World!', x, y);
```

Stroke Text

```
context.font = '40pt Arial';
context.strokeStyle = 'red';
context.strokeText('Hello World!', x, y);
```

Bold Text

```
context.font = 'bold 40px Arial';
```

Italic Text

```
context.font = 'italic 40px Arial';
```

Text Align

```
context.textAlign = 'start|end|left|center|right';
```

Text Baseline

```
context.textBaseline = 'top|hanging|middle|alphabetic|ideographic|bottom';
```

Get Text Width

```
var width = context.measureText('Hello world').width;
```

- **TRANSFORMS**

Translate

```
context.translate(x, y);
```

Scale

```
context.scale(x, y);
```

Rotate

```
context.rotate(radians);
```

Flip Horizontally

```
context.scale(-1, 1);
```

Flip Vertically

```
context.scale(1, -1);
```

Custom Transform

```
context.transform(a, b, c, d ,e, f);
```

Set Transform

```
context.setTransform(a, b, c, d ,e, f);
```

Shear

```
context.transform(1, sy, sx, 1, 0, 0);
```

Reset

```
context.setTransform(1, 0, 0, 1, 0, 0);
```

- **STATE STACK**

Push State onto Stack

```
context.save();
```

Pop State off of Stack

```
context.restore();
```

- **CLIPPING**

Clip

```
// draw path here
context.clip();
```

- **IMAGE DATA**

Get Image Data

```
var imageData = context.getImageData(x, y, width, height);
var data = imageData.data;
```

Loop Through Image Pixels

```
var imageData = context.getImageData(x, y, width, height);
var data = imageData.data;
var len = data.length;
var i, red, green, blue, alpha;

for(i = 0; i < len; i += 4) {
    red = data[i];
    green = data[i + 1];
    blue = data[i + 2];
    alpha = data[i + 3];
}
```

Loop Through Image Pixels With Coordinates

```
var imageData = context.getImageData(x, y, width, height);
var data = imageData.data;
var x, y, red, green, blue, alpha;

for(y = 0; y < imageHeight; y++) {
    for(x = 0; x < imageWidth; x++) {
        red = data[((imageWidth * y) + x) * 4];
        green = data[((imageWidth * y) + x) * 4 + 1];
        blue = data[((imageWidth * y) + x) * 4 + 2];
        alpha = data[((imageWidth * y) + x) * 4 + 3];
    }
}
```

Set Image Data

```
context.putImageData(imageData, x, y);
```

- **DATA URLS**

Get Data URL

```
var dataURL = canvas.toDataURL();
```

Render Canvas with Data URL

```
var imageObj = new Image();
imageObj.onload = function() {
    context.drawImage(imageObj, 0, 0);
};

imageObj.src = dataURL;
```

- **COMPOSITES**

Composite Operations

```
context.globalCompositeOperation = 'source-atop|source-in|source-out|source-over|destination-atop|destination-in|destination-out|destination-over|lighter|xor|copy';
```