Tools: Beliebiger Code-Editor (z.B. Visual Studio Code)
Autor: Leonard Hlavin
Letzte Änderung: 17.05.2021

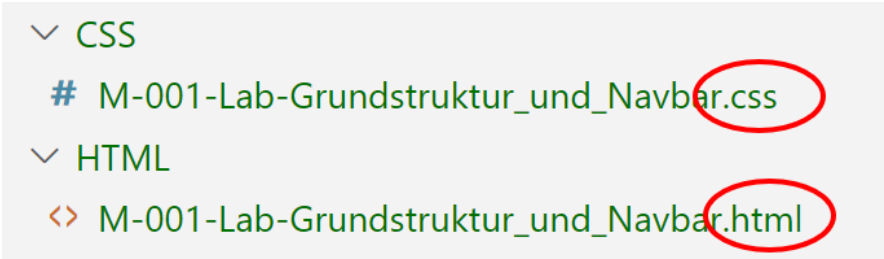
Modul 05 – Canvas

Ziel: Erstellen einer einfachen Grafik mit dem Canvas-Element

Vorbemerkung: Canvas ist ein HTML-Tag, das wir wie eine leere Leinwand (engl.: *canvas*) verwenden können, um mit JS darauf zu zeichnen.

Aufgabenstellung:

1. Erstellen Sie drei neue Files mit VS Code, ein HTML-File, ein CSS-File und ein JS-File und verknüpfen Sie diese miteinander. Sie können HTML-, CSS- und JS-Files zwecks Übersichtlichkeit auch in verschiedenen Ordnern ablegen. Achten Sie auf die korrekte File-Endung.



Hinweis: Zwecks leichterer Auffindbarkeit/Übersichtlichkeit befindet sich die Lösung in einem einzigen HTML-File (mit *script*- und *style*-Tag für JS und CSS). In der Realität trennen!

2. Erstellen Sie in Ihrem HTML-File ein 500x500 Pixel großes Canvas-Tag. Vergeben Sie eine ID, damit wir dieses Element über JS leicht ansprechen können.

HTML:

```
<canvas id="canvas-1" class="canvas-demo" width="500" height="500"></canvas>
```

Hinweise:

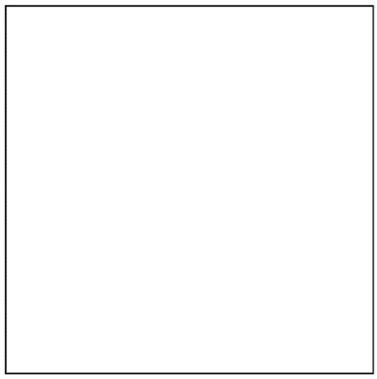
- Geben Sie bei *width* und *height* keine Einheit an. Gemeint sind damit automatisch Pixel.
- Sie können *width* und *height* auch über JS setzen.
- Canvas braucht ein schließendes Tag!

3. Geben Sie dem Canvas-Element mit CSS einen Rahmen, damit wir zumindest einmal sehen können, wo es sich befindet (und wie groß es ist).

CSS:

```
.canvas-demo{
    border: 2px solid black;
}
```

Ergebnis:



4. Erstellen Sie mit JS eine Variable für das Canvas-Element und setzen Sie den *context*.

JS:

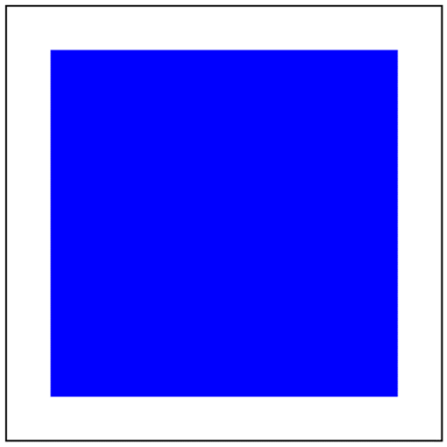
```
var canvas = document.getElementById('canvas-1');
var ctx = canvas.getContext('2d');
```

5. Zeichnen Sie ein Quadrat und zentrieren sie es. Verwenden Sie eine beliebige Hintergrundfarbe (im Beispiel unten blau).

JS:

```
ctx.fillStyle = 'rgb(0, 0, 255)';
ctx.fillRect(50, 50, 400, 400);
```

Ergebnis:



Hinweis: Die ersten beiden Werte geben x- und y-Koordinate für die linke obere Ecke Ihres Rechteckes an, die nächsten beiden Werte stehen für *width* und *height*.

6. Befüllen Sie nun das blaue Quadrat mit 4x4 weiteren Quadraten in anderen Farben, so dass zuletzt alles blau verdeckt ist. Sie können theoretisch 16 einzelne Quadrate erstellen – oder zwei miteinandern verschachtelte Schleifen verwenden.

JS:

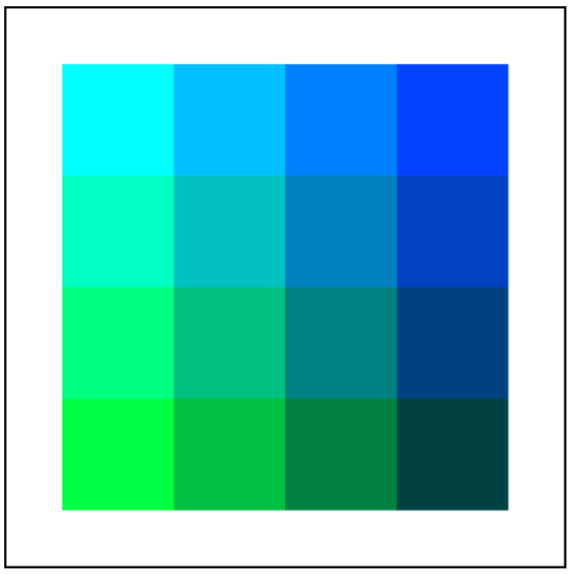
Variante mit 16 einzelnen Quadraten (davon hier die ersten zwei):

```
// 1
ctx.fillStyle = 'rgb(0, 255, 255)';
ctx.fillRect(50, 50, 100, 100);
// 2
ctx.fillStyle = 'rgb(0, 192, 255)';
ctx.fillRect(150, 50, 100, 100);
```

Die elegantere und schnellere Variante mit zwei verschachtelten for-Schleifen:

```
for(var i = 0; i < 4; i++){
    for(var j = 0; j < 4; j++){
        ctx.fillStyle = 'rgb(0,' + Math.floor(255 - j * 63) + ', '
                                 + Math.floor(255 - i * 63) + ')';
        ctx.fillRect(j*100+50, i*100+50, 100, 100);
    }
}
```

Ergebnis:



7. Zeichnen Sie mittig über die Quadrate drei konzentrische Kreise.

JS:

```
// Kreis 1
  ctx.beginPath();
  ctx.arc(250, 250, 150, 0, Math.PI*2, false);
  ctx.fill();
```

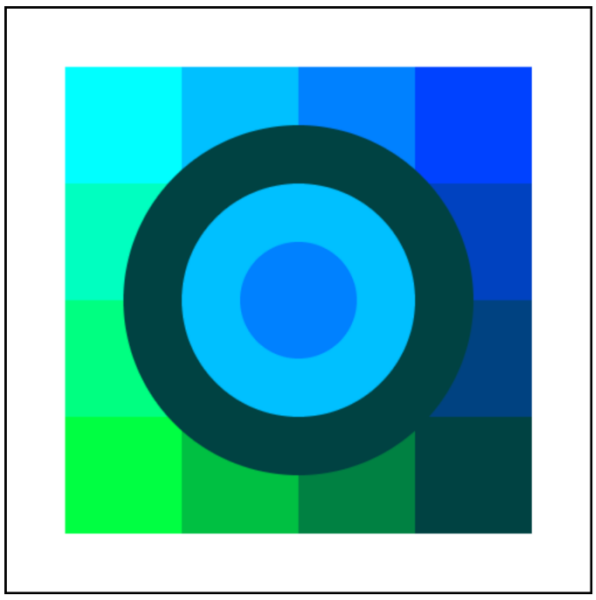
```
// Kreis 2
  ctx.fillStyle = 'rgb(0, 192, 255)';
  ctx.beginPath();
  ctx.arc(250, 250, 100, 0, Math.PI*2, false);
  ctx.fill();

// Kreis 3
  ctx.fillStyle = 'rgb(0, 129, 255)';
  ctx.beginPath();
  ctx.arc(250, 250, 50, 0, Math.PI*2, false);
  ctx.fill();
```

Hinweis:

Wenn Sie für den ersten Kreis keinen fillStyle (Farbe) definieren, dann wird die zuletzt verwendete Farbe auch auf dieses Element angewendet (in unserem Fall, die letzte Farbe, die für unsere kleinen Quadrate berechnet wurde).

Ergebnis:



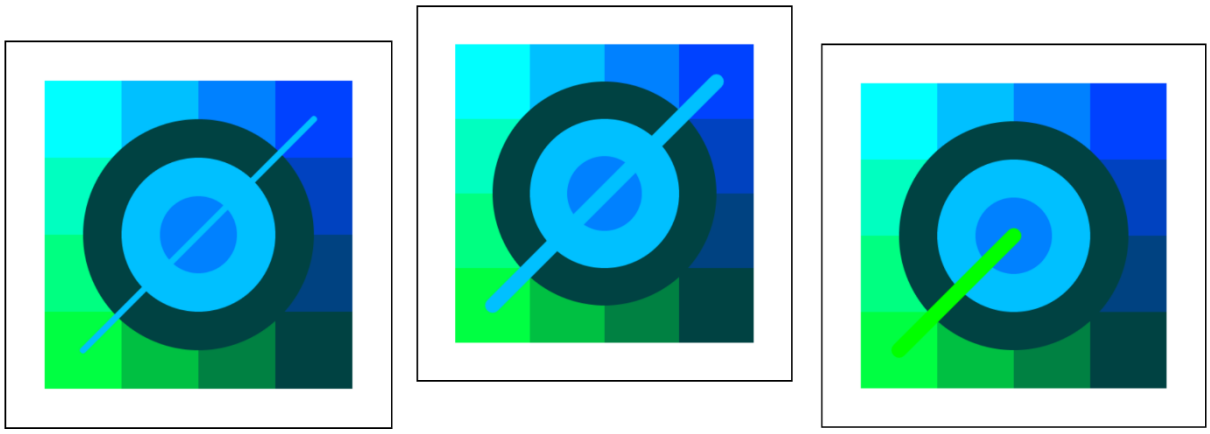
8. Ziehen Sie eine diagonale Line von rechts oben nach links unten über die Kreise. Die Linie soll sich nicht über die gesamte Diagonale erstrecken, sondern etwa in der Hälfte der äußeren Quadrate beginnen. Die Enden sollen abgerundet sein.

   Experimentieren Sie mit verschiedenen Farben. Sie können spannende Effekte erzeugen, wenn Sie eine Farbe wählen, die im Hintergrund schon vorkommt. Auch die Breite der diagonalen Linie macht einen großen Unterschied.

JS:

```
// diagonaler Strich
        ctx.strokeStyle = 'rgb(0, 192, 255)';
        ctx.lineWidth = 20;
        ctx.lineCap = 'round';
        ctx.beginPath();
        ctx.moveTo(400, 100);
        ctx.lineTo(100, 400);
        ctx.stroke();
```

Beispiele:



Hinweis: Die ursprüngliche blaue Hintergrundfarbe sehen wir nicht mehr und könnten sie entfernen – oder sie z.B. versetzt anzeigen: `ctx.fillRect(40, 40, 400, 400);`