

Farbregel-Generator

```
#p5sketch { position: relative; width: 724px; height: 910px; margin-left: 0; } canvas { position: absolute; left: 0; top: 0; z-index: 0; } .custom-slider { position: absolute; -webkit-appearance: none; height: 10px; background: #888; border-radius: 5px; outline: none; z-index: 1; } .custom-slider::-webkit-slider-thumb { -webkit-appearance: none; width: 16px; height: 16px; background: #fff; border-radius: 50%; cursor: pointer; border: 2px solid #000; } .custom-slider::-moz-range-thumb { width: 16px; height: 16px; background: #fff; border-radius: 50%; cursor: pointer; border: 2px solid #000; }
```

Erklärung der Farbregel

Eine Farbregel beschreibt Farben nicht einzeln, sondern als Zahlen: Ein **Start-Winkel** (Basis-Hue, 0-359°) legt die erste Farbe fest, ein **Versatz** (in Grad auf dem Farbkreis) legt jede weitere Farbe fest. Sättigung und Helligkeit bleiben konstant, sodass allein der Farbton entscheidet.

Der Generator berechnet Versätze, Hue-Werte, HSB- und Hex-Werte und gibt den Code für die eigene Farbregel aus, den man direkt in einen p5.js-Sketch kopieren kann.

Farbharmonien als Winkelbeziehungen:

- komplementär: 0° und 180° (gegenüberliegend)
- analog: -30°, 0°, +30° (benachbart)
- triadisch: 0°, 120°, 240° (gleichmäßig verteilt)
- tetradisch: 0°, 60°, 180°, 240° (zwei Komplementärpaare)

Zur Kontrolle der Lesbarkeit: Mit der Graustufen-Ansicht lässt sich prüfen, ob die Farben auch ohne Farbsehen unterscheidbar bleiben. Der Grauwert ist die Luminanz der Farbe (Gewichtung nach WCAG), der Kontrast bezieht sich auf einen weißen Hintergrund. Gleiche Helligkeit im HSB-Modell ergibt je nach Farbton sehr unterschiedliche Grauwerte.

Siehe auch [additive Farbmischung \(RGB\)](#) und [subtraktive Farbmischung \(CMY\)](#).

Beispiel p5.js Code

```
// Farbregel-Generator (p5.js)
// Start-Winkel, Harmonie, Sättigung und Helligkeit einstellen.
// Der Sketch zeigt Versatz, Hue, HSB- und Hex-Werte und erzeugt den Code für die eigene Farbregel.

let baseSlider, satSlider, briSlider, harmSelect, grayBox, codeBox;
let ring;

const HARMONIEN = {
  "komplementär": [0, 180],
  "analog": [-30, 0, 30],
  "triadisch": [0, 120, 240],
  "tetradisch": [0, 60, 180, 240]
};

function setup() {
  let canvas = createCanvas(724, 580);
```

```
canvas.parent("p5sketch");
noLoop();

// Regler und Canvas liegen im Container "p5sketch" (Positionen relativ dazu)
baseSlider = createSlider(0, 359, 20);
baseSlider.parent("p5sketch");
baseSlider.class("custom-slider");
baseSlider.position(20, 62);
baseSlider.style("width", "250px");

harmSelect = createSelect();
harmSelect.parent("p5sketch");
for (let name in HARMONIEN) harmSelect.option(name);
harmSelect.selected("triadisch");
harmSelect.position(20, 142);
harmSelect.style("width", "250px");

satSlider = createSlider(0, 100, 80);
satSlider.parent("p5sketch");
satSlider.class("custom-slider");
satSlider.position(20, 212);
satSlider.style("width", "250px");

briSlider = createSlider(0, 100, 90);
briSlider.parent("p5sketch");
briSlider.class("custom-slider");
briSlider.position(20, 282);
briSlider.style("width", "250px");

grayBox = createCheckbox(" Graustufen-Ansicht", false);
grayBox.parent("p5sketch");
grayBox.position(20, 326);

for (let el of [baseSlider, satSlider, briSlider, harmSelect, grayBox]) {
  el.input() => redraw();
  el.changed() => redraw();
}

// Farbkreis einmal vorrendern (0° oben, im Uhrzeigersinn)
ring = createGraphics(200, 200);
ring.noFill();
ring.strokeWeight(24);
ring.strokeCap(SQUARE);
for (let h = 0; h < 360; h += 0.5) {
  let c = hsb2rgb(h, 100, 100);
  ring.stroke(c[0], c[1], c[2]);
  ring.arc(100, 100, 152, 152, radians(h - 90), radians(h - 90 + 1.2));
}

// Code zum Kopieren (markierbarer Text unter dem Canvas)
codeBox = createElement("pre", "");
codeBox.parent("p5sketch");
codeBox.style("font-family", "monospace");
codeBox.style("font-size", "13px");
codeBox.style("background", "#f3f4f6");
codeBox.style("padding", "12px");
```

```
codeBox.style("width", "684px");
codeBox.style("box-sizing", "border-box");
codeBox.style("overflow-x", "auto");
codeBox.style("user-select", "text");
}

function draw() {
  background(255);

  const basis = baseSlider.value();
  const sat = satSlider.value();
  const bri = briSlider.value();
  const name = harmSelect.value();
  const versatz = HARMONIEN[name];
  const grau = grayBox.checked();

  // Beschriftung der Regler
  noStroke();
  fill(90);
  textFont("sans-serif");
  textSize(12);
  textStyle(BOLD);
  text("START-WINKEL (HUE)", 20, 52); fill(30); textStyle(NORMAL); textSize(14);
  text(basis + "°", 230, 52);
  fill(90); textStyle(BOLD); textSize(12);
  text("HARMONIE", 20, 132);
  text("SÄTTIGUNG", 20, 202); fill(30); textStyle(NORMAL); textSize(14);
  text(sat, 245, 202);
  fill(90); textStyle(BOLD); textSize(12);
  text("HELLIGKEIT", 20, 272); fill(30); textStyle(NORMAL); textSize(14);
  text(bri, 245, 272);

  // Farbkreis mit Markern
  const cx = 150, cy = 470;
  image(ring, cx - 100, cy - 100);
  let pts = [];
  for (let i = 0; i < versatz.length; i++) {
    let h = norm360(basis + versatz[i]);
    let a = radians(h - 90);
    pts.push([cx + 76 * cos(a), cy + 76 * sin(a)]);
  }
  stroke(30);
  strokeWeight(2);
  noFill();
  for (let i = 0; i < pts.length; i++) {
    let j = (i + 1) % pts.length;
    if (pts.length === 2 && i === 1) break;
    line(pts[i][0], pts[i][1], pts[j][0], pts[j][1]);
  }
  for (let i = 0; i < versatz.length; i++) {
    let h = norm360(basis + versatz[i]);
    let c = farbeAnzeige(h, sat, bri, grau);
    stroke(30); strokeWeight(2);
    fill(255);
    circle(pts[i][0], pts[i][1], versatz[i] === 0 ? 24 : 20);
    noStroke();
  }
}
```

```
    fill(c[0], c[1], c[2]);
    circle(pts[i][0], pts[i][1], versatz[i] === 0 ? 17 : 13);
  }
  noStroke(); fill(30); textAlign(CENTER, CENTER); textSize(22); textStyle(BOLD);
  text(basis + "°", cx, cy - 4);
  textSize(12); textStyle(NORMAL); fill(90);
  text(name, cx, cy + 16);
  textAlign(LEFT, BASELINE);

// Karten mit den berechneten Farben
const kartenX = 300, kartenY = 30, kb = 98, abstand = 8;
const codeHex = [], codeHue = [];
for (let i = 0; i < versatz.length; i++) {
  let h = norm360(basis + versatz[i]);
  let rgb = hsb2rgb(h, sat, bri);
  let g = grauwert(rgb);
  let x = kartenX + i * (kb + abstand);

  stroke(215); strokeWeight(1); fill(248);
  rect(x, kartenY, kb, 190, 8);
  let c = farbeAnzeige(h, sat, bri, grau);
  noStroke(); fill(c[0], c[1], c[2]);
  rect(x + 1, kartenY + 1, kb - 2, 80, 7, 7, 0, 0);

  fill(30); textStyle(BOLD); textSize(15);
  text(vorzeichen(versatz[i]), x + 8, kartenY + 104);
  textStyle(NORMAL); textSize(11); fill(80);
  text("Hue " + h + "°", x + 8, kartenY + 124);
  text("HSB " + h + " " + sat + " " + bri, x + 8, kartenY + 140);
  text(hexCode(rgb), x + 8, kartenY + 156);
  text("Grau " + g + " " + nf(kontrast(rgb), 1, 1) + ":1", x + 8, kartenY +
172);

  codeHex.push(hexCode(rgb));
  codeHue.push(h);
}

// Hinweis
fill(110); textSize(11);
text("Grau = Luminanz (WCAG). Kontrast bezieht sich auf weißen Hintergrund.",
kartenX, kartenY + 212);

// Code aktualisieren
codeBox.html(
  "// Farbregel: Basis-Hue + Versatz je Gruppe (" + name + ")\n" +
  "// in setup(): colorMode(HSB, 360, 100, 100);\n\n" +
  "const basisHue = " + basis + ";\n" +
  "const versatz = [" + versatz.join(", ") + "];\n" +
  "const sat = " + sat + ";\n" +
  "const bri = " + bri + ";\n\n" +
  "function farbe(gruppe) {\n" +
  "  const h = (basisHue + versatz[gruppe % versatz.length] + 360) % 360;\n" +
  "  return color(h, sat, bri);\n" +
  "}\n\n" +
  "// Kontrolle: Hue " + codeHue.join(", ") + "\n" +
  "// Hex " + codeHex.join(", ")
```

```
);
codeBox.position(20, 592);
}

// ----- Hilfsfunktionen -----
function norm360(h) { return ((h % 360) + 360) % 360; }

function vorzeichen(o) {
  if (o === 0) return "Basis";
  return (o > 0 ? "+" : "-") + abs(o) + "°";
}

function hsb2rgb(h, s, b) {
  h = norm360(h); s /= 100; let v = b / 100;
  let c = v * s, x = c * (1 - abs((h / 60) % 2 - 1)), m = v - c, r = 0, g = 0, bl = 0;
  if (h < 60) { r = c; g = x; }
  else if (h < 120) { r = x; g = c; }
  else if (h < 180) { g = c; bl = x; }
  else if (h < 240) { g = x; bl = c; }
  else if (h < 300) { r = x; bl = c; }
  else { r = c; bl = x; }
  return [round((r + m) * 255), round((g + m) * 255), round((bl + m) * 255)];
}

function hexCode(rgb) {
  return "#" + rgb.map(v => v.toString(16).padStart(2, "0")).join("").toUpperCase();
}

function lin(c) { c /= 255; return c <= 0.04045 ? c / 12.92 : pow((c + 0.055) / 1.055, 2.4); }
function luminanz(rgb) { return 0.2126 * lin(rgb[0]) + 0.7152 * lin(rgb[1]) + 0.0722 * lin(rgb[2]); }
function grauwert(rgb) {
  let y = luminanz(rgb);
  let v = y <= 0.0031308 ? 12.92 * y : 1.055 * pow(y, 1 / 2.4) - 0.055;
  return constrain(round(v * 255), 0, 255);
}

function kontrast(rgb) { return 1.05 / (luminanz(rgb) + 0.05); }

function farbeAnzeige(h, s, b, grau) {
  let rgb = hsb2rgb(h, s, b);
  if (!grau) return rgb;
  let g = grauwert(rgb);
  return [g, g, g];
}
```

From:
<https://wiki.ct-lab.info/> - **Creative Technologies Lab** | dokuWiki

Permanent link:
<https://wiki.ct-lab.info/doku.php/extras:codikon:p5js:farbregel-generator>

Last update: **2026/10/20 06:09**



