pixels

An array containing the color of each pixel on the canvas.

Colors are stored as numbers representing red, green, blue, and alpha (RGBA) values. `pixels` is a one-dimensional array for performance reasons.

Each pixel occupies four elements in the `pixels` array, one for each RGBA value. For example, the pixel at coordinates (0, 0) stores its RGBA values at `pixels[0]`, `pixels[1]`, `pixels[2]`, and `pixels[3]`, respectively. The next pixel at coordinates (1, 0) stores its RGBA values at `pixels[4]`, `pixels[5]`, `pixels[6]`, and `pixels[7]`. And so on. The `pixels` array for a 100×100 canvas has 100 × 100 × 4 = 40,000 elements.

Some displays use several smaller pixels to set the color at a single point. The pixelDensity() function returns the pixel density of the canvas. High density displays often have a pixelDensity() of 2. On such a display, the `pixels` array for a 100×100 canvas has 200 × 200 × 4 = 160,000 elements.

Accessing the RGBA values for a point on the canvas requires a little math as shown below. The loadPixels() function must be called before accessing the `pixels` array. The updatePixels() function must be called after any changes are made.

Examples

```
function setup() {
  createCanvas(100, 100);

  // Load the pixels array.
  loadPixels();

  // Set the dot's coordinates.
  let x = 50;
  let y = 50;

  // Get the pixel density.
  let d = pixelDensity();

  // Set the pixel(s) at the center of the canvas black.
  for (let i = 0; i < d; i += 1) {
    for (let j = 0; j < d; j += 1) {
      let index = 4 * ((y * d + j) * width * d + (x * d + i));
      // Red.
      pixels[index] = 0;
      // Green.
      pixels[index + 1] = 0;
      // Blue.
      pixels[index + 2] = 0;
      // Alpha.
      pixels[index + 3] = 255;
    }
  }

  // Update the canvas.
  updatePixels();
```

```
function setup() {
  createCanvas(100, 100);

  // Load the pixels array.
  loadPixels();

  // Get the pixel density.
  let d = pixelDensity();

  // Calculate the halfway index in the pixels array.
  let halfImage = 4 * (d * width) * (d * height / 2);

  // Make the top half of the canvas red.
  for (let i = 0; i < halfImage; i += 4) {
    // Red.
    pixels[i] = 255;
    // Green.
    pixels[i + 1] = 0;
    // Blue.
    pixels[i + 2] = 0;
    // Alpha.
    pixels[i + 3] = 255;
  }

  // Update the canvas.
  updatePixels();

  describe('A red rectangle drawn above a gray rectangle.');
}
```

```
function setup() {
  createCanvas(100, 100);

  // Create a p5.Color object.
  let pink = color(255, 102, 204);

  // Load the pixels array.
  loadPixels();

  // Get the pixel density.
  let d = pixelDensity();

  // Calculate the halfway index in the pixels array.
  let halfImage = 4 * (d * width) * (d * height / 2);

  // Make the top half of the canvas red.
  for (let i = 0; i < halfImage; i += 4) {
    pixels[i] = red(pink);
    pixels[i + 1] = green(pink);
    pixels[i + 2] = blue(pink);
    pixels[i + 3] = alpha(pink);
  }

  // Update the canvas.
  updatePixels();

  describe('A pink rectangle drawn above a gray rectangle.');
```

This page is generated from the comments in src/image/pixels.js . Please feel free to edit it and submit a pull request!

Related References

blend
Copies a region of pixels from one image to another.

copy
Copies pixels from a source image to a region of the canvas.

filter
Applies an image filter to the canvas.

get
Gets a pixel or a region of pixels from the canvas.