filter()

Applies an image filter to the canvas.

The preset options are:

`INVERT` Inverts the colors in the image. No parameter is used.

`GRAY` Converts the image to grayscale. No parameter is used.

`THRESHOLD` Converts the image to black and white. Pixels with a grayscale value above a given threshold are converted to white. The rest are converted to black. The threshold must be between 0.0 (black) and 1.0 (white). If no value is specified, 0.5 is used.

`OPAQUE` Sets the alpha channel to entirely opaque. No parameter is used.

`POSTERIZE` Limits the number of colors in the image. Each color channel is limited to the number of colors specified. Values between 2 and 255 are valid, but results are most noticeable with lower values. The default value is 4.

`BLUR` Blurs the image. The level of blurring is specified by a blur radius. Larger values increase the blur. The default value is 4. A gaussian blur is used in `P2D` mode. A box blur is used in `WEBGL` mode.

`ERODE` Reduces the light areas. No parameter is used.

`DILATE` Increases the light areas. No parameter is used.

`filter()` uses WebGL in the background by default because it's faster. This can be disabled in `P2D` mode by adding a `false` argument, as in `filter(BLUR, false)`. This may be useful to keep computation off the GPU or to work around a lack of WebGL support.

In WebGL mode, `filter()` can also use custom shaders. See createFilterShader() for more information.

Examples

```
let img;

// Load the image.
function preload() {
  img = loadImage('/assets/bricks.jpg');
}

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

  // Display the image.
  image(img, 0, 0);

  // Apply the INVERT filter.
  filter(INVERT);

  describe('A blue brick wall.');
}
```

```
let img;

// Load the image.
function preload() {
  img = loadImage('/assets/bricks.jpg');
}

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

  // Display the image.
  image(img, 0, 0);

  // Apply the GRAY filter.
  filter(GRAY);

  describe('A brick wall drawn in grayscale.');
}
```

```
let img;

// Load the image.
function preload() {
  img = loadImage('/assets/bricks.jpg');
}

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

  // Display the image.
  image(img, 0, 0);

  // Apply the THRESHOLD filter.
  filter(THRESHOLD);

  describe('A brick wall drawn in black and white.');
}
```

```
let img;

// Load the image.
function preload() {
  img = loadImage('/assets/bricks.jpg');
}

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

  // Display the image.
  image(img, 0, 0);

  // Apply the OPAQUE filter.
  filter(OPAQUE);

  describe('A red brick wall.');
}
```

```
let img;

// Load the image.
function preload() {
  img = loadImage('/assets/bricks.jpg');
}

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

  // Display the image.
  image(img, 0, 0);

  // Apply the POSTERIZE filter.
  filter(POSTERIZE, 3);

  describe('An image of a red brick wall drawn with limited color palette.');
}
```

```
let img;

// Load the image.
function preload() {
  img = loadImage('/assets/bricks.jpg');
}

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

  // Display the image.
  image(img, 0, 0);

  // Apply the BLUR filter.
  filter(BLUR, 3);

  describe('A blurry image of a red brick wall.');
}
```

```
let img;

// Load the image.
function preload() {
  img = loadImage('/assets/bricks.jpg');
}

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

  // Display the image.
  image(img, 0, 0);

  // Apply the DILATE filter.
  filter(DILATE);

  describe('A red brick wall with bright lines between each brick.');
}
```

```
let img;

// Load the image.
function preload() {
  img = loadImage('/assets/bricks.jpg');
}

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

  // Display the image.
  image(img, 0, 0);

  // Apply the ERODE filter.
  filter(ERODE);

  describe('A red brick wall with faint lines between each brick.');
}
```

```
let img;

// Load the image.
function preload() {
  img = loadImage('/assets/bricks.jpg');
}

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

  // Display the image.
  image(img, 0, 0);

  // Apply the BLUR filter.
  // Don't use WebGL.
  filter(BLUR, 3, false);

  describe('A blurry image of a red brick wall.');
}
```

Syntax

```
filter(filterType, [filterParam], [useWebGL])
```

```
filter(filterType, [useWebGL])
```

```
filter(shaderFilter)
```

Parameters

filterType	Constant: either THRESHOLD, GRAY, OPAQUE, INVERT, POSTERIZE, BLUR, ERODE, DILATE or BLUR.
filterParam	Number: parameter unique to each filter.
useWebGL	Boolean: flag to control whether to use fast WebGL filters (GPU) or original image filters (CPU); defaults to `true`.
shaderFilter	p5.Shader: shader that's been loaded, with the frag shader using a `tex0` uniform.

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.