createFramebuffer()

Creates and a new p5.Framebuffer object.

p5.Framebuffer objects are separate drawing surfaces that can be used as textures in WebGL mode. They're similar to p5.Graphics objects and generally run much faster when used as textures.

The parameter, `options`, is optional. An object can be passed to configure the p5.Framebuffer object. The available properties are:

- `format`: data format of the texture, either `UNSIGNED_BYTE`, `FLOAT`, or `HALF_FLOAT`. Default is `UNSIGNED_BYTE`.
- `channels`: whether to store `RGB` or `RGBA` color channels. Default is to match the main canvas which is `RGBA`.
- `depth`: whether to include a depth buffer. Default is `true`.
- `depthFormat`: data format of depth information, either `UNSIGNED_INT` or `FLOAT`. Default is `FLOAT`.
- `stencil`: whether to include a stencil buffer for masking. `depth` must be `true` for this feature to work. Defaults to the value of `depth` which is `true`.
- `antialias`: whether to perform anti-aliasing. If set to `true`, as in `{ antialias: true }`, 2 samples will be used by default. The number of samples can also be set, as in `{ antialias: 4 }`. Default is to match setAttributes() which is `false` (`true` in Safari).
- `width`: width of the p5.Framebuffer object. Default is to always match the main canvas width.
- `height`: height of the p5.Framebuffer object. Default is to always match the main canvas height.
- `density`: pixel density of the p5.Framebuffer object. Default is to always match the main canvas pixel density.
- `textureFiltering`: how to read values from the p5.Framebuffer object. Either `LINEAR` (nearby pixels will be interpolated) or `NEAREST` (no interpolation). Generally, use `LINEAR` when using the texture as an image and `NEAREST` if reading the texture as data. Default is `LINEAR`.

If the `width`, `height`, or `density` attributes are set, they won't automatically match the main canvas and must be changed manually.

Note: `createFramebuffer()` can only be used in WebGL mode.

Examples

```
let myBuffer;

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

  // Create a p5.Framebuffer object.
  myBuffer = createFramebuffer();

  describe('A grid of white toruses rotating against a dark
gray background.');
}

function draw() {
  background(50);

  // Start drawing to the p5.Framebuffer object.
  myBuffer.begin();

  // Clear the drawing surface.
  clear();

  // Turn on the lights.
  lights();

  // Rotate the coordinate system.
  rotateX(frameCount * 0.01);
  rotateY(frameCount * 0.01);

  // Style the torus.
  noStroke();

  // Draw the torus.
  torus(20);

  // Stop drawing to the p5.Framebuffer object.
```

```
let myBuffer;

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

  // Create an options object.
  let options = { width: 25, height: 25 };

  // Create a p5.Framebuffer object.
  // Use options for configuration.
  myBuffer = createFramebuffer(options);

  describe('A grid of white toruses rotating against a dark
gray background.');
}

function draw() {
  background(50);

  // Start drawing to the p5.Framebuffer object.
  myBuffer.begin();

  // Clear the drawing surface.
  clear();

  // Turn on the lights.
  lights();

  // Rotate the coordinate system.
  rotateX(frameCount * 0.01);
  rotateY(frameCount * 0.01);
```

Syntax

```
createFramebuffer([options])
```

Parameters

options	Object: configuration options.

Returns

p5.Framebuffer: new framebuffer.

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

Related References

createFramebuffer
Creates a new p5.Framebuffer object with the same WebGL context as the graphics buffer.

remove
Removes the graphics buffer from the web page.

reset
Resets the graphics buffer's transformations and lighting.

blendMode
Sets the way colors blend when added to the canvas.