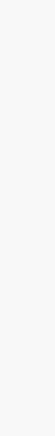
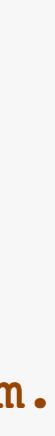


# reset()

Resets the graphics buffer's transformations and lighting.

By default, the main canvas resets certain transformation and lighting values each time `draw()` executes. `p5.Graphics` objects must reset these values manually by calling `myGraphics.reset()`.

## Examples



```
let pg;

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

  // Create a p5.Graphics object.
  pg = createGraphics(60, 60);

  describe('A white circle moves downward slowly within a dark square. The circle resets at the top of the dark square when the user presses the mouse.');
}

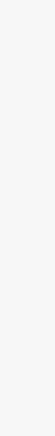
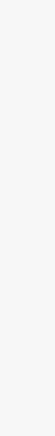
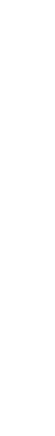
function draw() {
  background(200);

  // Translate the p5.Graphics object's coordinate system.
  // The translation accumulates; the white circle moves.
  pg.translate(0, 0.1);

  // Draw to the p5.Graphics object.
  pg.background(100);
  pg.circle(30, 0, 10);

  // Display the p5.Graphics object.
  image(pg, 20, 20);

  // Translate the main canvas' coordinate system.
  // The translation doesn't accumulate; the dark
```



```
let pg;

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

  // Create a p5.Graphics object.
  pg = createGraphics(60, 60);

  describe('A white circle at the center of a dark gray square. The image is drawn on a light gray background.');
}

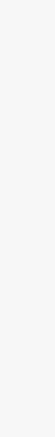
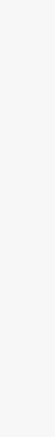
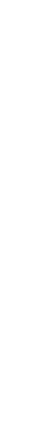
function draw() {
  background(200);

  // Translate the p5.Graphics object's coordinate system.
  pg.translate(30, 30);

  // Draw to the p5.Graphics object.
  pg.background(100);
  pg.circle(0, 0, 10);

  // Display the p5.Graphics object.
  image(pg, 20, 20);

  // Reset the p5.Graphics object automatically.
  pg.reset();
}
```



```
let pg;

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

  // Create a p5.Graphics object using WebGL mode.
  pg = createGraphics(100, 100, WEBGL);

  describe("A sphere lit from above with a red light. The sphere's surface becomes glossy while the user clicks and holds the mouse.");
}

function draw() {
  background(200);

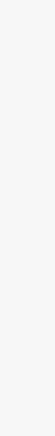
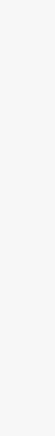
  // Add a red point light from the top-right.
  pg.pointLight(255, 0, 0, 50, -100, 50);

  // Style the sphere.
  // It should appear glossy when the
  // lighting values are reset.
  pg.noStroke();
  pg.specularMaterial(255);
  pg.shininess(100);

  // Draw the sphere.
  pg.sphere(30);

  // Display the p5.Graphics object.
  image(pg, 0, 0);

  // Reset the p5.Graphics object automatically.
  pg.reset();
}
```



```
let pg;

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

  // Create a p5.Graphics object using WebGL mode.
  pg = createGraphics(100, 100, WEBGL);

  describe('A sphere with a glossy surface is lit from the top-right by a red light.');
}

function draw() {
  background(200);

  // Add a red point light from the top-right.
  pg.pointLight(255, 0, 0, 50, -100, 50);

  // Style the sphere.
  pg.noStroke();
  pg.specularMaterial(255);
  pg.shininess(100);

  // Draw the sphere.
  pg.sphere(30);

  // Display the p5.Graphics object.
  image(pg, 0, 0);

  // Reset the p5.Graphics object automatically.
  pg.reset();
}
```

This page is generated from the comments in `src/core/p5.Graphics.js`. Please feel free to edit it and submit a pull request!

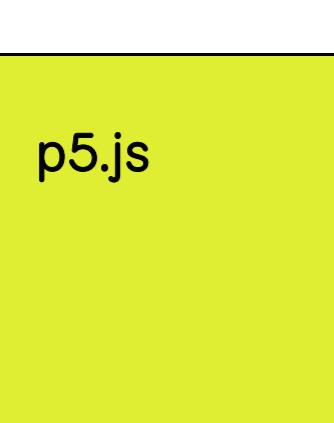
## Related References

`createFramebuffer()`  
Creates a new `p5.Framebuffer` object with the same `p5.js` 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.



Donate Today! Support p5.js and the Processing Foundation.

