Detecting Mouse Events

15 min

p5.js also provides us with functions that trigger when specific *mouse events* are detected. Some examples of mouse events include pressing the mouse, pressing and releasing the mouse, and detecting mouse movements. The mouse event functions in p5.js work like JavaScript DOM events behind the scenes, in which p5.js waits for an event to happen to run specific mouse event functions.

The `mousePressed()` function is called once after every mouse button press over the canvas. This means that the code block within the `mousePressed()` function will only run once the mouse has been pressed. To run the code again, the mouse needs to be released and pressed a second time.

We can also use the built-in `mouseIsPressed` variable to determine whether the mouse is pressed or not. When the mouse is pressed, the `mouseIsPressed` variable evaluates to `true`, and when it is not pressed, it evaluates to `false`. We can create an `if` statement using the `mouseIsPressed` variable to continuously run a code block while the mouse is pressed.

Take a look at **mouseEvents.js** on the right. The code shows the differences in using the `mousePressed()` function and the `mouseIsPressed` variable. Once a mouse press is detected, the `mouseIsPressed` variable returns `true` and begins drawing

ellipses at random positions around the canvas for as long as the mouse is pressed. Simultaneously, the `mousePressed()` function randomly chooses the color of the ellipses and the background, but those colors will not change until the mouse is released and pressed again.

There are also other mouse event functions that p5.js offers, such as:

- The `mouseMoved()` function, which can be used to trigger an event every time the mouse moves while the mouse has not been pressed.
- The `mouseClicked()` function, which can be used to detect a mouse button press and release over an element.

Some mouse events act in similar ways, but they can be layered to create various interactions within a p5.js project.

Instructions

1.

In the browser window notice the ellipse in the center of the canvas. Currently, it's pretty bland—let's make it more interesting!

Open **sketch.js** and add an `if` statement into the `draw()` function. It should have a condition of `mouseIsPressed`. Inside the code block of the `if` statement, set the fill value for the ellipse below to:

```
fill(fillValue, 0, 0, 50);
```

Add an `else` statement to the `if` statement. Inside the code block of the `else` statement set fill to:

```
fill(0, 0, fillValue, 50);
```

Note that the `fillValue` variable used in the if-else statement is declared at the top of the program and initialized with a value of `0` in the `setup()` function.
Hint
Remember that `mouseIsPressed` is a built-in variable and has the following syntax when being used in an `if` statement:

```
if (mouseIsPressed) {
  // this block of code runs if the mouse is pressed
} else {
  // this block of code runs if the mouse NOT is pressed
}
```

2.

After the `draw()` function, create an empty `mouseMoved()` function. Inside the function, set the `fillValue` variable to a random value between 0 and 255.

Now move your mouse around the canvas to see the shape's color change!
Hint
To generate a random number between 0 and 255 for the variable, the maximum range given to the `random()` function should be 256 rather than 255. Remember that the randomly generated number does not include the number given as the maximum range itself.

3.

After the `mouseMoved()` function block, at the bottom of the program, create an empty `mousePressed()` function.
Hint
The function's syntax should look something like this:

```
function mousePressed() {
  //this code block will run when mouse is pressed
}
```

4.

Using the `shapeScale` variable declared at the top of the program, double the size of the ellipse each time the mouse is clicked.

To begin, inside the empty `mousePressed()` function, write an `if` statement with a condition of:

```
shapeScale < width
```

If the statement evaluates to `true`, assign `shapeScale` to be twice its current value. Do this using a multiplication assignment operator (`*=`).

If the statement evaluates to `false`, the `shapeScale` variable should be reset to 50.

Now, click on the canvas and see how the ellipse's size doubles each time the canvas is clicked. When the size of the ellipse becomes larger than the width of the canvas it is reset back to its original size of 50.
Hint
The `mousePressed()` function should look something like this:

```
 function mousePressed() {
if (shapeScale < width) {
  shapeScale *= 2;
```

```
}
else{
  shapeScale = 50;
}
}
```

mouseEvents.js

```
let randR = 0;
let randG = 0;
let randB = 0;

function setup(){
  createCanvas(windowWidth, windowHeight);
  background(255);
}

function draw() {
  // Code continuously runs when the mouse is held down
  if (mouseIsPressed) {
    let x = random(width);
    let y = random(height);
    fill(randR, randG, randB);
    ellipse(x, y, 16, 16);
  }
}

function mousePressed() {
  clear();
  // Code only runs once each mouse press.
  randR = random(256);
  randG = random(256);
  randB = random(256);
  background(mouseX % 256, mouseY % 256, randB);
}
```

sketch.js

```
let fillValue; // Used to modify the ellipse's fill color
let shapeScale; // Used to scale the size of the ellipse

function setup() {
```

```
  createCanvas(windowWidth, windowHeight);
  noStroke();

  fillValue = 0;
  shapeScale = 50;
}

function draw() {
  background(75, 50);

  // TODO: Add if statement to check if mouseIsPressed and set fill colors accordingly
  if (mouseIsPressed) {
    fill(fillValue, 0, 0, 50);
  } else {
    fill(0, 0, fillValue, 50);
  }

  ellipse(width / 2, height / 2, shapeScale, shapeScale);
}

// TODO: Create a mouseMoved() function and inside the function, set fillValue to a
random number between 0 and 255
function mouseMoved() {
  fillValue = random(0, 255);
}


// TODO: Create a mousePressed() function and inside the function, modify shapeScale
variable
function mousePressed() {
  if(shapeScale < width) {
    shapeScale *= 2;
  } else {
    shapeScale = 50;
  }
}
```