

# if

A way to choose whether to run a block of code.

`if` statements are helpful for running a block of code based on a condition. For example, an `if` statement makes it easy to express the idea "Draw a circle if the mouse is pressed":

```
if (mouseIsPressed === true) {
  circle(mouseX, mouseY, 20);
}
```

The statement header begins with the keyword `if`. The expression in parentheses `mouseIsPressed === true` is a Boolean expression that's either `true` or `false`. The code between the curly braces `{}` is the `if` statement's body. The body only runs if the Boolean expression is `true`.

The `mouseIsPressed` system variable is always `true` or `false`, so the code snippet above could also be written as follows:

```
if (mouseIsPressed) {
  circle(mouseX, mouseY, 20);
}
```

An `if-else` statement adds a block of code that runs if the Boolean expression is `false`. For example, here's an `if-else` statement that expresses the idea "Draw a circle if the mouse is pressed. Otherwise, display a message":

```
if (mouseIsPressed === true) {
  // When true.
  circle(mouseX, mouseY, 20);
} else {
  // When false.
  text('Click me!', 50, 50);
}
```

There are two possible paths, or branches, in this code snippet. The program must follow one branch or the other.

An `else-if` statement makes it possible to add more branches. `else-if` statements run different blocks of code under different conditions. For example, an `else-if` statement makes it easy to express the idea "If the mouse is on the left, paint the canvas white. If the mouse is in the middle, paint the canvas gray. Otherwise, paint the canvas black":

```
if (mouseX < 33) {
  background(255);
} else if (mouseX < 67) {
  background(200);
} else {
  background(0);
}
```

`if` statements can add as many `else-if` statements as needed. However, there can only be one `else` statement and it must be last.

`if` statements can also check for multiple conditions at once. For example, the Boolean operator `&&` (AND) checks whether two expressions are both `true`:

```
if (keyIsPressed === true && key === 'p') {
  text('You pressed the "p" key!', 50, 50);
}
```

If the user is pressing a key AND that key is `'p'`, then a message will display.

The Boolean operator `||` (OR) checks whether at least one of two expressions is `true`:

```
if (keyIsPressed === true || mouseIsPressed === true) {
  text('You did something!', 50, 50);
}
```

If the user presses a key, or presses a mouse button, or both, then a message will display.

The body of an `if` statement can contain another `if` statement. This is called a "nested `if` statement." For example, nested `if` statements make it easy to express the idea "If a key is pressed, then check if the key is `'r'`. If it is, then set the fill to red":

```
if (keyIsPressed === true) {
  if (key === 'r') {
    fill('red');
  }
}
```

See [Boolean](#) and [Number](#) to learn more about these data types and the operations they support.

## Examples

```
// Click the mouse to show the circle.

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

  describe(
    'A white circle on a gray background. The circle follows the mouse user clicks on the canvas.'
  );
}

function draw() {
  background(200);

  if (mouseIsPressed === true) {
    circle(mouseX, mouseY, 20);
  }
}
```

  

```
// Click the mouse to show different shapes.

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

  describe(
    'A white ellipse on a gray background. The ellipse becomes a circle when the user presses the mouse.'
  );
}

function draw() {
  background(200);

  if (mouseIsPressed === true) {
    circle(50, 50, 20);
  } else {
    ellipse(50, 50, 20, 50);
  }
}
```

  

```
// Move the mouse to change the background color.

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

  describe(
    'A square changes color from white to black as the user moves the mouse from left to right.'
  );
}

function draw() {
  if (mouseX < 33) {
    background(255);
  } else if (mouseX < 67) {
    background(200);
  } else {
    background(0);
  }
}
```

  

```
// Click on the canvas to begin detecting key presses.

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

  describe(
    'A white circle drawn on a gray background. The circle changes color to red when the user presses the "r" key.'
  );
}

function draw() {
  background(200);

  if (keyIsPressed === true) {
    if (key === 'r') {
      fill('red');
    }
  }

  circle(50, 50, 40);
}
```

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

## Related References

[class](#)  
A template for creating objects of a particular type.

[console](#)  
Prints a message to the web browser's console.

[for](#)  
A way to repeat a block of code when the number of iterations is known.

[function](#)  
A named group of statements.

[p5.js](#)

[Resources](#)

[Information](#)

[Socials](#)

[Reference](#)

[Download](#)

[GitHub ↗](#)

[Tutorials](#)

[Contact](#)

[Instagram ↗](#)

[Examples](#)

[Copyright](#)

[X ↗](#)

[Contribute](#)

[Privacy Policy](#)

[YouTube ↗](#)

[Community](#)

[Terms of Use](#)

[Discord ↗](#)

[About](#)

[Forum ↗](#)

[Start Coding](#)

[Donate](#)

[Forum ↗](#)



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

✖