function

A named group of statements.

Functions help with organizing and reusing code. For example, functions make it easy to express the idea "Draw a flower.":

```
function drawFlower() {
  // Style the text.
  textAlign(CENTER, CENTER);
  textSize(20);

  // Draw a flower emoji.
  text('🌸', 50, 50);
}
```

The function header begins with the keyword `function`. The function's name, `drawFlower`, is followed by parentheses `()` and curly braces `{}`. The code between the curly braces is called the function's body. The function's body runs when the function is called like so:

```
drawFlower();
```

Functions can accept inputs by adding parameters to their headers. Parameters are placeholders for values that will be provided when the function is called. For example, the `drawFlower()` function could include a parameter for the flower's size:

```
function drawFlower(size) {
  // Style the text.
  textAlign(CENTER, CENTER);

  // Use the size parameter.
  textSize(size);

  // Draw a flower emoji.
  text('🌸', 50, 50);
}
```

Parameters are part of the function's declaration. Arguments are provided by the code that calls a function. When a function is called, arguments are assigned to parameters:

```
// The argument 20 is assigned to the parameter size.
drawFlower(20);
```

Functions can have multiple parameters separated by commas. Parameters can have any type. For example, the `drawFlower()` function could accept `Number` parameters for the flower's x- and y-coordinates along with its size:

```
function drawFlower(x, y, size) {
  // Style the text.
  textAlign(CENTER, CENTER);

  // Use the size parameter.
  textSize(size);

  // Draw a flower emoji.
  // Use the x and y parameters.
  text('🌸', x, y);
}
```

Functions can also produce outputs by adding a `return` statement:

```
function double(x) {
  let answer = 2 * x;
  return answer;
}
```

The expression following `return` can produce an output that's used elsewhere. For example, the output of the `double()` function can be assigned to a variable:

```
let six = double(3);
text(`3 x 2 = ${six}`, 50, 50);
```

Examples

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

  describe('A pink flower on a gray background.');
}

function draw() {
  background(200);

  // Call the drawFlower() function.
  drawFlower();
}

// Declare a function that draws a flower at the
// center of the canvas.
function drawFlower() {
  // Style the text.
  textAlign(CENTER, CENTER);
  textSize(20);

  // Draw a flower emoji.
  text('🌸', 50, 50);
}
```

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

  describe('A pink flower on a gray background.');
}

function draw() {
  background(200);

  // Call the drawFlower() function and pass values for
  // its position and size.
  drawFlower(50, 50, 20);
}

// Declare a function that draws a flower at the
// center of the canvas.
function drawFlower(x, y, size) {
  // Style the text.
  textAlign(CENTER, CENTER);

  // Use the size parameter.
  textSize(size);

  // Draw a flower emoji.
  // Use the x and y parameters.
  text('🌸', x, y);
}
```

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

  describe('The message "Hello, 🌍!" written on a gray
background.');
}

function draw() {
  background(200);

  // Create a greeting.
  let greeting = createGreeting('🌍');

  // Style the text.
  textAlign(CENTER, CENTER);
  textSize(16);

  // Display the greeting.
  text(greeting, 50, 50);
}

// Return a string with a personalized greeting.
function createGreeting(name) {
  let message = `Hello, ${name}!`;
  return message;
}
```

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.