class

A template for creating objects of a particular type.

Classes can make it easier to program with objects. For example, a `Frog` class could create objects that behave like frogs. Each object created using a class is called an instance of that class. All instances of a class are the same type. Here's an example of creating an instance of a `Frog` class:

```
let fifi = new Frog(50, 50, 20);
```

The variable `fifi` refers to an instance of the `Frog` class. The keyword `new` is used to call the `Frog` class' constructor in the statement `new Frog()`. Altogether, a new `Frog` object was created and assigned to the variable `fifi`. Classes are templates, so they can be used to create more than one instance:

```
// First Frog instance.
let frog1 = new Frog(25, 50, 10);

// Second Frog instance.
let frog2 = new Frog(75, 50, 10);
```

A simple `Frog` class could be declared as follows:

```
class Frog {
  constructor(x, y, size) {
    // This code runs once when an instance is created.
    this.x = x;
    this.y = y;
    this.size = size;
  }

  show() {
    // This code runs once when myFrog.show() is called.
    textAlign(CENTER, CENTER);
    textSize(this.size);
    text('🐸', this.x, this.y);
  }

  hop() {
    // This code runs once when myFrog.hop() is called.
    this.x += random(-10, 10);
    this.y += random(-10, 10);
  }
}
```

Class declarations begin with the keyword `class` followed by the class name, such as `Frog`, and curly braces `{}`. Class names should use `PascalCase` and can't have spaces in their names. For example, naming a class `Kermit The Frog` with spaces between each word would throw a `SyntaxError`. The code between the curly braces `{}` defines the class.

Functions that belong to a class are called methods. `constructor()`, `show()`, and `hop()` are methods in the `Frog` class. Methods define an object's behavior. Methods can accept parameters and return values, just like functions. Note that methods don't use the `function` keyword.

`constructor()` is a special method that's called once when an instance of the class is created. The statement `new Frog()` calls the `Frog` class' `constructor()` method.

A class definition is a template for instances. The keyword `this` refers to an instance's data and methods. For example, each `Frog` instance has unique coordinates stored in `this.x` and `this.y`. The `show()` method uses those coordinates to draw the frog. The `hop()` method updates those coordinates when called. Once a `Frog` instance is created, its data and methods can be accessed using the dot operator `.` as follows:

```
// Draw a lily pad.
fill('green');
stroke('green');
circle(fifi.x, fifi.y, 2 * fifi.size);

// Show the Frog.
fifi.show();

// Hop.
fifi.hop();
```

Examples

```
// Declare a frog variable.
let fifi;

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

  // Assign the frog variable a new Frog object.
  fifi = new Frog(50, 50, 20);

  describe('A frog face drawn on a blue background.');
}

function draw() {
  background('cornflowerblue');

  // Show the frog.
  fifi.show();
}

class Frog {
  constructor(x, y, size) {
    this.x = x;
    this.y = y;
    this.size = size;
  }

  show() {
    textAlign(CENTER, CENTER);
    textSize(this.size);
    text('🐸', this.x, this.y);
  }
}
```

```
// Declare two frog variables.
let frog1;
let frog2;

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

  // Assign the frog variables a new Frog object.
  frog1 = new Frog(25, 50, 10);
  frog2 = new Frog(75, 50, 20);

  describe('Two frog faces drawn next to each other on a blue background.');
}

function draw() {
  background('cornflowerblue');

  // Show the frogs.
  frog1.show();
  frog2.show();
}

class Frog {
  constructor(x, y, size) {
    this.x = x;
    this.y = y;
    this.size = size;
  }

  show() {
    textAlign(CENTER, CENTER);
    textSize(this.size);
    text('🐸', this.x, this.y);
  }
}
```

```
// Declare two frog variables.
let frog1;
let frog2;

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

  // Assign the frog variables a new Frog object.
  frog1 = new Frog(25, 50, 10);
  frog2 = new Frog(75, 50, 20);

  // Slow the frame rate.
  frameRate(1);

  describe('Two frog faces on a blue background. The frogs hop around randomly.');
}

function draw() {
  background('cornflowerblue');

  // Show the frogs.
  frog1.show();
  frog2.show();

  // Move the frogs.
  frog1.hop();
  frog2.hop();

  // Wrap around if they've hopped off the edge.
  frog1.checkEdges();
  frog2.checkEdges();
}
```

```
// Create an array that will hold frogs.
let frogs = [];

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

  // Add Frog objects to the array.
  for (let i = 0; i < 5; i += 1) {
    // Calculate random coordinates and size.
    let x = random(0, 100);
    let y = random(0, 100);
    let s = random(2, 20);

    // Create a new Frog object.
    let frog = new Frog(x, y, s);

    // Add the Frog to the array.
    frogs.push(frog);
  }

  // Slow the frame rate.
  frameRate(1);

  describe(
    'Five frog faces on a blue background. The frogs hop around randomly.'
  );
}

function draw() {
```

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.