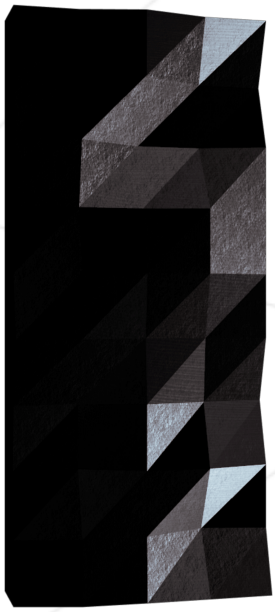


APCSP Part 5

Objects and Prototypes in Javascript

Kai kai@42.us.org

Summary: The second part of the Abstraction concept is about being able to create your own objects.

Contents

Chapter I

Multiple Levels of Abstraction

Functions and Objects are the two important abstraction tools that you must learn how to use. In addition to these building blocks, there are some other general ideas about what abstraction does that you should keep in mind.

I.1 Abstraction in Programming Languages

Different programming languages offer different levels of abstraction. You may be familiar with the kids' coding language Scratch or Snap, where colorful building blocks allow you to create variables, loops and if statements via drag and drop. That is a high abstraction language, also called a `"high-level language"`. Most popular, "coding is fun!" languages are high level languages, including Javascript and Python (altough not as abstract as Scratch.) It's high abstraction because the *complex details are hidden from view.*

If you join the 42 adult curriculum or try out a Taste of the C Piscine on your own, you will start to learn a low abstraction programming language, known as a `low level language`. C is considerably less abstract than Javascript because it contains a lot of tools for directly manipulating the bits of data in a computer's memory. In order to be useful for programming on tiny microprocessors, C does not hide complex details from the programmer. You have to keep track of more details about how the bits are moving around in memory in order to program in C.

Even lower than C are languages on the level of Assembly. These days, it's not necessary for many people to write Assembly code, but other programing languages are translated into Assembly by the computer before they are run.

I.2 Abstraction in Data Storage

As always remember that all of the data stored in your programs is saved inside of physcial, metal and plastic electronic circuit boards - as bits of data. So, the images, strings, and numbers we work with are built on top of abstractions that hide the raw 1010101011010 binary from us. Just for fun: Adam Savage, Bits and Bytes

Chapter II

Javascript's Weirdness

If you have previously learned another language up to the point of using "objects" and "classes" in that language, you need to know from the very start: **Javascript is Different.** Most mainstream programming languages follow a pattern of object oriented programming that is based on `Classes`. Javascript follows a pattern of object oriented programming that is based on `Prototypes`.

Much of the hatred of Javascript exists because people only learn class-based OOP in college, and then they never truly understand how prototype-based OOP works. But you are lucky, you are growing up with Javascript and so you can become jedi masters of both. :)

If you are coming from experience with object-oriented programming in Java, Python, Ruby, C++, Pascal or PHP, start by reading this blog post to orient yourself in a new, prototype-based way.

Chapter III

The Object Concept

III.1 Everything is an Object, except Primitives

III.1.1 Primitives

Objects are used whenever you need do define a "thing" that has behavior more complex than the basic types. Here's a recap of the six primitive types in Javascript:

Number: any type of number, including what other languages would call "ints" (integers) and "floats" (decimal points).

String: any amount of text, from a single letter to a paragraph.

Boolean: a true/false flag that can be toggled back and forth.

Null: represents a "zero" or an empty string.

Undefined: any variable that has been declared but has no value yet.

Symbol: a string that has become a language keyword.

Don't really worry about the last three right now, although you should remember that "null" means "nothing". The point is that there are a few basic primitives, but every other "thing" in Javascript is an Object.

III.1.2 These are a Few of My Favorite Objects

Some of the built-in objects we (will) use in p5js are:

Number: Okay so it's a primitive, but there is also an object version, which has more complex behavior than the primitive type allows. Mozilla Documentation

String: Just like number, string is an object too... Mozilla Documentation

Boolean: And again, the plot thickens... Mozilla Documentation

Array: Keep the documentation handy so you know how to add to and remove from arrays. Mozilla Documentation

Color: Remember R, G, B? p5js Documentation

Image: In p5js, you can create an Image object from a picture that you upload, and modify its pixels. p5js Documentation

Vector: Once you are ready for them, vectors can make your dynamic games much more powerful. p5js Documentation

Dictionary: If you are familiar with dictionaries in Python, know that Javascript has various versions of them too. p5js Documentation

Table: We'll use the p5js Table objects later in our data visualization project.

III.2 Build your Own Objects!

Objects allow you to encapsulate `variables` and `functions` that "go together", partly so that it's easier to remember what variables exist in your code, and what they are called.

Chapter IV

Object Literals

Here is an example of the same program written without using objects, and then with using objects. Open the two side by side.

Read the comments as well, and play the game. What parts of the code are the same, and what parts are different?

The second sketch is using "object literal" syntax, the most straightforward way of creating an object. Use this syntax when you just need to make one object at a time.

```
// Object literal syntax
var sprite = {
  x: 0,
  y: 0,
  c: 'black',
  radius: 10,
}
```

All you have to do is declare a variable (var myVar = ...) and then start a curly-brace block. Inside the curly braces, use "property: value" syntax to add variables that belong to the object.

Later on in the code, you can access the properties of that object with dot syntax.

```
    sprite.x = random(width); // Sets the x variable inside of sprite to a random number between 0 and width
    sprite.y = random(height); // Sets the y variable inside of sprite
```

The first benefit we get from packaging several variables as an object is that we can send all of the "sprite" information into the draw_character function as just one parameter.

```
// Now that we have objects, this function only needs one parameter.
function draw_character(name){
  fill(name.c);
  ellipse(name.x, name.y, name.radius, name.radius);
}
draw_character(sprite);
```

Before, it was like this:

```
// Requires four parameters
function draw_character(x, y, c, radius){
  fill(c);
  ellipse(x, y, radius, radius);
}
draw_character(sprite_x, sprite_y, sprite_color, sprite_radius);
```

You can even define a function that belongs to the object, as shown in my example with target.teleport().

```
// Object literal with a function ability
var target = {
  x: 0,
  y: 0,
  c: 'orange',
  radius: 10,
  teleport: function(x, y) {
          target.x = x;
          target.y = y;
        }
}
target.teleport(random(width), random(height));
```

This is like creating a character called "target" and then giving him a special ability, "teleport". He also has characteristics - his location, size, and color.

Read this page of the Mozilla documentation: Basics of Objects.

IV.1 HW1: Check your Understanding

In this sketch, I have all the parts for a glorified text box (speech bubble) which wiggles around the canvas. I have deleted the part of the code where the object, speechBubble, is defined. Recreate the speechBubble object so that draw_speechBubble() works with no modification.

Chapter V

Constructor Functions

V.1 Example

Here is another version of the example program, this time using a constructor function to fill up an array of 20 target dots. Open the two side by side.

Read the comments, and study the differences. What changed to make it create a whole army of dot clones (without writing the same code 20 times)?

Here is the anatomy of how a constructor fuction works:

```
// Constructor function for target dots
function Target(x, y, c, r) {
  this.x = x;
  this.y = y;
  this.c = c;
  this.radius = r;
  this.teleport = function(x, y) {
    this.x = x;
    this.y = y;
  }
}

// Inside the setup() function, I used a for loop to do this many times:
var target = new Target(277, 33, #a96836, 10);
```

It's the "new" keyword that allows you to use a function as a constructor. When you write a constructor function, you should name it with a capital letter ("Target") to help people who read your code understand what it is. The "this" keyword is a symbol that points to the object that is currently begin created or used. Since a constructor function can be used over and over to create many similar objects, "this" refers to the current copy that is being created.

The return value of a constructor function is an object that was newly created. You can store it any any variable or any array of variables.

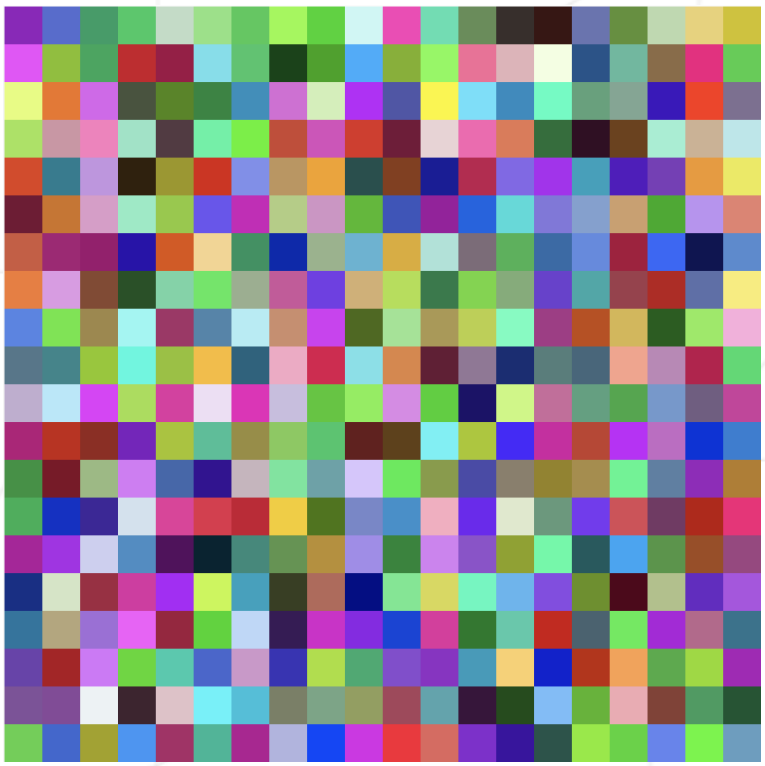
V.2 Reading

If you want more time to think about these concepts, I recommend watching and coding along to these two videos: The Coding Train 7.3, Arrays of Objects; and The Coding

Train 7.4, The Constructor Function in Javascript .

V.3 HW2: Practice

Create a sketch which has a grid of squares (like the chessboard). Each square should be an object, and they are all created from the same constructor function. Each square should have a random color which is generated by choosing a random green, random red, and random blue value. Store the squares in a double array (an array inside an array), and display them to the sketch.

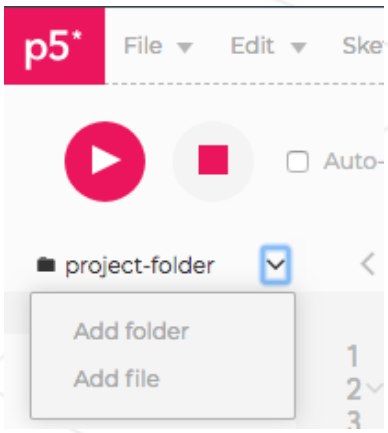


Chapter VI

Playing with Images

Adding images to your sketches is fun! You can either use them as background pictures, or pieces of the game.

To add images in the p5js web editor, expand the sidebar on the left (there's an arrow under the Play button, left of "sketch.js"), and then click the down arrow next to "project-folder" to access the menu for adding an image. You will want to upload files from your computer, so download them first.



JPEG images do not work inside of the p5js editor; if you get an error, try opening your image in Preview and exporting it as a PNG.

Use CTRL+F to search the p5js reference for functions related to images, and read about them.

VI.1 Example

Here's an example of the CodeForFun owl, Cody, dressed up as a pirate. He appears a random point on the screen and when you press space, shoots cannonballs in random directions!

VI.2 HW3: Try it yourself

Modify the sketch from before so that the sprite which we control with the keyboard, is a sprite image of your choice.

Hint: You may want to give the sprite and the targets different display functions now.

Make sure the sprite still grows in size when it eats dots!