Splash of Code

Cracked

Learn JavaScript by Making Computer Art

Joel Dare

Copyright © 2019

Forward

My first computer was a TRS-80 Color Computer from Radio Shack. It wasn't new, my Mom had purchased it second hand. It would connect to a TV through an analog antenna connection.

When you turned it on you were greeted with a green screen and the word, “OK”. In order to do anything you had to feed the computer a program, which you did by simply typing it in. Luckily, it came with a few books and these books had program listings for cool utilities and some games. You’d spend a few hours typing in a program then type “run” and it would jump to life.

When you turned off the computer, later, it would forget what you had typed and you'd have to start the process over the next day. This repetition, of typing other people's programs, is how I learned to code. Over time, the code I was typing in started to make sense to me and I'd use reference books to fill in the gaps as I learned.

I created the Splash of Code series to try to bring that joy from my childhood to you. I was lucky to start young, when my mind was suppel, but you can learn programming at any age. Learning this craft has brought me extreme joy and I hope to share that joy with you.

Introduction

Splash of Code is a series of short guides that teaches computer programming by walking you through the creation of art projects. We'll be working in JavaScript, a language that works in your web browser with no additional tools. Our goal is to create a simple image inspired by cracks in a concrete wall.

I've provided the complete source code for this project so you can simply type it from the code listing here. By doing so you'll end up with a finished art piece that's unique and that you can print, frame, and display. You'll also start to learn some JavaScript programming. You don't need to understand everything right away. Just relax and copy the code. You'll be creating great artwork and you'll start to learn a few things as you go along.

Learning to code requires a lot of patience and repetition. You won't understand everything by creating this one project. In fact, your first few projects may be more frustrating than fun. Keep trying. The learning comes over time.

If you dedicate yourself to creating a dozen projects, you'll start to understand the key concepts. The more you search, study, and evaluate, the more you'll learn. Before you know it you'll find yourself considering improvements and alternatives for the code you're reading and writing.

Getting Started

You'll need a computer with a web browser. Nothing else is required. I've created a basic HTML file for you on *CodePen*, a tool that lets you write code online. Open the URL below to get started.

https://codepen.io/codazoda/pen/aPVGGx

Start at the *Complete Code* section and re-type the code into CodePen. Once you've typed the program turn to the *Code Walk Through* and read it. There I'll explain each of the small sections in more detail. Don't worry if you don't understand most of it. Understanding it comes with repetition. You can also go back and forth between the *Code Walkthrough* and the *Computer Code* for a little overview of anything you want more information about. Then search the internet for more detailed explanations.

The project consists of two files. An HTML file that you won't need to edit and a JavaScript file. You'll retype the code in this guide into that JavaScript file. More information about the files is available at *https://splashofcode.com/template*.

As you work on this project you'll probably run into typos that create errors. If the image doesn't draw, or doesn't draw in the way you expect, double check that you typed everything correctly and look at the output in the developer console to see if there are any errors. Search the internet for instructions on how to open the developer console in your particular web browser.

Once you've created an image, you can save it by right clicking and selecting *Save Image As*.

Complete Code

```
// Setup some params
var width = 4 * 300;
var height = 6 * 300;
var lineLen = 50;
var halfLineLen = lineLen / 2;

// Grab the canvas element from the page
var myCanvas =
     document.getElementById("myCanvas");

// Set the canvas to the calculated size
myCanvas.width = width;
myCanvas.height = height;

// Grab the canvas to draw on
var ctx = myCanvas.getContext("2d");

// Begin a path
ctx.beginPath();

// Set the line width
ctx.lineWidth = 5;

// Draw the first line
x = width / 2;
y = height / 2;
ctx.moveTo(x, y - halfLineLen);
```

```
ctx.lineTo(x, y + halfLineLen);
lastX = x;
lastY = y + halfLineLen;
ctx.stroke();

// Loop and draw more lines
let linesToDraw = 2000;
let linesDrawn = 0;
let draw = setInterval(function(){
     let xPos = -1;
     let yPos = -1;
     // Pick a direction until it's in
     while (xPos < 0 ||
     xPos > width ||
     yPos < 0 ||
     yPos > height) {
     let lastAngle = 180;
     let deg = rand(0, 360);
     xPos = lastX +
          lineLen *
          Math.cos(
               Math.PI * deg / 180.0
          );
     yPos = lastY +
          lineLen *
          Math.sin(
               Math.PI * deg / 180.0
          );
     }
     ctx.lineTo(xPos, yPos);
```

```
      lastX = xPos;
      lastY = yPos;
      ctx.stroke();
      linesDrawn++;
      // Stop if we've drawn enough lines
      if (linesDrawn >= linesToDraw) {
      clearInterval(draw);
      }
}, 10);

// End the path
ctx.closePath();

// Pick a random number
function rand(min, max) {
      min = Math.ceil(min);
      max = Math.floor(max);
      let rand = Math.random();
      return Math.floor(rand *
      (max - min + 1)) +
      min;
}
```

You're a Programmer

If you ran that code and you got it to work, you've just completed a program. You are a programmer.

In the 1980s, when the personal computer was in its infancy, we copied code from books and magazines. As a side effect, many of us learned computer programming, almost by accident. At the time, programming, or at least typing in other peoples programs, was necessary to use a computer at all. Storage, like hard drives and disk drives, were rare and code listings were often published in print.

One of the best ways to learn computer programming is to copy the work of others. It's how many of today's senior engineers learned and how they continue learning. Professional developers copy code nearly every day of their careers. If you're an absolute beginner or if you've struggled to understand programming concepts from a class, it's a great way for you to learn too.

Stack Overflow is one of the most popular websites for exactly this reason. Copying code isn't such a stigma in computer science. Your goal is to create a program. You can build your first, using other

people's code, with almost no knowledge of how it works. Over time you'll start to understand bits and pieces of the code you type.

You do need to be careful about copyright if you plan to use your new code publicly. There is a lot of free and open source code to borrow from though. Developers share code so that others can use it and learn from it. We collaborate to compare, refine, and improve. As you get curious you can dig deeper by searching for specific things that you want to understand better. Most likely, you'll find someone else who's already examined the topic and has explained it to others in great detail.

Learning to code takes time, to be sure, but copying code from others is a great way to get started.

Programming is also a life-long learning exercise. Once you start to learn, you'll continue to learn for as long as you pursue the craft. There's a never-ending supply of learning. Computer languages are being expanded regularly and there are tons of languages to learn. If you have the patience for it, it's a great way to exercise your brain.

Code Walkthrough

First, we setup a few parameters.

```
// Setup some params
var width = 4 * 300;
var height = 6 * 300;
var lineLen = 50;
var halfLineLen = lineLen / 2;
```

Now we grab the canvas element from the HTML page. Notice that this code is wrapped onto two lines. This is a limitation of the size of this guide. JavaScript lets you make wider lines but I've wrapped this line at the equals sign. That works fine but it might be a little unusual. Compare it to the lines above.

```
// Grab the canvas element from the page
var myCanvas =
     document.getElementById("myCanvas");
```

Now that we have the canvas we need to set its width and height based on the initial calculations we made above.

```
// Set the canvas to the calculated size
myCanvas.width = width;
```

```
myCanvas.height = height;
```

Now we grab the *context* of the canvas. We'll do all our actual drawing on the context that we've assigned.

```
// Grab the canvas to draw on
var ctx = myCanvas.getContext("2d");
```

Next we tell the canvas that we're going to begin a path, we set a line width, and we draw our first line.

```
// Begin a path
ctx.beginPath();

// Set the line width
ctx.lineWidth = 5;

// Draw the first line
x = width / 2;
y = height / 2;
ctx.moveTo(x, y - halfLineLen);
ctx.lineTo(x, y + halfLineLen);
lastX = x;
lastY = y + halfLineLen;
ctx.stroke();
```

Here, we create a loop to draw more lines starting at the end of our first line and moving in a random

direction from 0 to 360 degrees. We're doing some math to figure out the length of that line, in pixels, based on the direction we picked. As we complete one line we pick a new direction and draw the next.

```
// Loop and draw more lines
let linesToDraw = 2000;
let linesDrawn = 0;
let draw = setInterval(function(){
     let xPos = -1;
     let yPos = -1;
     // Pick a direction until it's in
     while (xPos < 0 ||
     xPos > width ||
     yPos < 0 ||
     yPos > height) {
     let lastAngle = 180;
     let deg = rand(0, 360);
     xPos = lastX +
          lineLen *
          Math.cos(
               Math.PI * deg / 180.0
          );
     yPos = lastY +
          lineLen *
          Math.sin(
               Math.PI * deg / 180.0
          );
     }
```

```
      ctx.lineTo(xPos, yPos);
      lastX = xPos;
      lastY = yPos;
      ctx.stroke();
      linesDrawn++;
      // Stop if we've drawn enough lines
      if (linesDrawn >= linesToDraw) {
      clearInterval(draw);
      }
}, 10);
```

All of the lines have been drawn so we close the path that we started earlier.

```
// End the path
ctx.closePath();
```

Finally, here's the function we used that generates random numbers for us.

```
// Pick a random number
function rand(min, max) {
      min = Math.ceil(min);
      max = Math.floor(max);
      let rand = Math.random();
      return Math.floor(rand *
      (max - min + 1)) +
      min;
}
```

Understanding Code Comments

A *comment* is an explanation or annotation in the source code of your program. They're added to make the code easier to understand but they're ignored by the interpreter.

A code comment in JavaScript starts with either two slashes or with a slash and a star.

You might use two slashes for a single line comment like the following.

```
// This code creates amazing art
```

You might also use a slash and a star to begin a comment that will run across multiple lines. You end the comment with a star and a slash (the opposite combination that you used to start the comment). Here's an example of a multi-line comment.

```
/* This program creates a drawing on an
HTML canvas. It draws 2,000 small dashes.
For each dash it picks a new, random,
direction. If the dash would end up
outside the bounds of the drawing page, it
picks again. */
```

It’s important to comment your code so that it’s easier to understand in the future.

Additional Reading

Read my guide, *How to Code,* for a quick introduction to JavaScript. That will help you with a basic understanding of the JavaScript console, data types, variables, math, and functions. You may find the background information there helpful.

The *How to Code* guide is available as part of the complete *Splash of Code* series at the URL below.

https://gum.co/splashofcode

Final Words

Thanks for reading *Cracked*, *Issue 5* in the *Splash of Code* series. If you enjoyed this guide, please encourage your friends or followers to subscribe in order to get their own free copy at the URL below.

https://splashofcode.com

If you have any questions or comments, feel free to email me directly. My email address is:

joel@splashofcode.com

In *Cracked* you'll start to learn programming by creating a piece of minimalist abstract computer art by retyping code from this guide.