**Uses of loops**

In this reading, we'll discuss, at a very high level, the reasons to use loops in JavaScript.

Note that we will keep this discussion high-level because there are multiple "pieces of the puzzle" that are still missing from your understanding at this point.

This is why we will not get bogged-down in the detail of syntax and implementation, but instead, simply discuss how and why loops are used in everyday work of JavaScript developers.

Consider the following example: You work as a developer for an online store.

The store is selling letter cubes for toddlers, and the entire "Shop now" section of the site is organized in a layout where each cube on sale is displayed in a simple card component, with an image of the cube, the letter it teaches, a short description, and the price.

Cards are organized in rows, so that each row contains three cards - three different letters.

Each card is a preview of that specific letter cube on sale, and it's also a link to an entire page, dedicated to providing more info about the cubes, their teaching value, and providing the visitor with a way to complete their checkout process.

Now, here's a quick question: where would loops fit into displaying this grid of cards showcasing the letter cubes on sale?

To understand just how this works, let me code a basic prototype of how this might work.

Since you still don't have enough knowledge to display website layouts in browser with the help of JavaScript, for now I'll have to settle for using a simple string and the console.

Still, this should be a fun exercise.

1

2

3

4

5

6

var cubes = 'ABCDEFG';

//styling console output using CSS with a %c format specifier

for (var i = 0; i < cubes.length; i++) {

    var styles = "font-size: 40px; border-radius: 10px; border: 1px solid blue; background: pink; color: purple";

    console.log("%c" + cubes[i], styles)

}

RunReset

***Note: In order to have the styles applied, try running this code snippet in your browser's console.***

That's it, with this simple code, the output in the console shows each letter on a separate line, styled like a letter cube for toddlers.

The code itself should be mostly familiar, except for the **cubes.length** and the **cubes[i]** syntax.

Without getting into too many details, here are both code snippets explained as simple as possible.

The **cubes.length** returns a number. Since **cubes** is a string of characters the **cubes.length** gives me the length of the string saved in the variable.

So this gives me the number 7, effectively making my for loop look like this:

1

2

3

4

5

6

var cubes = 'ABCDEFG';

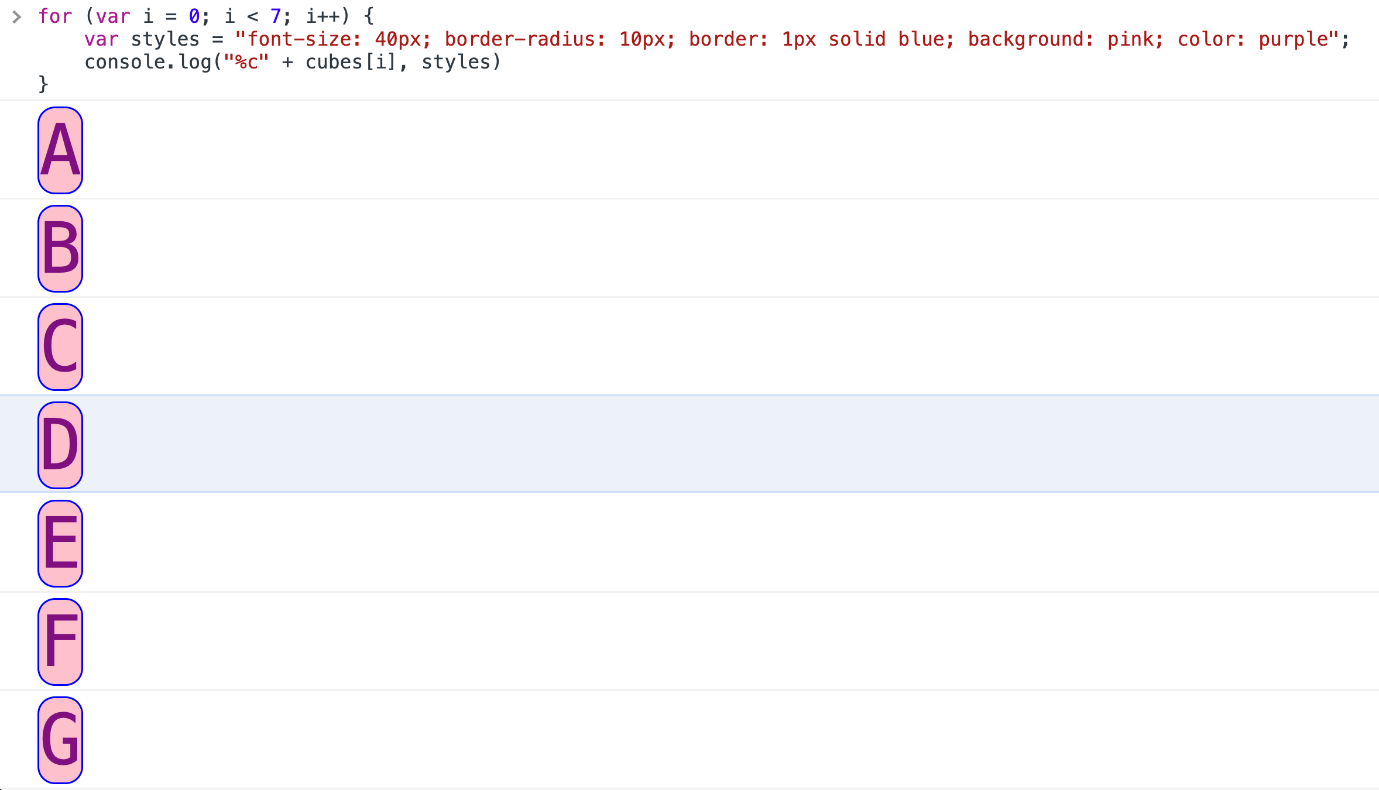
//styling console output using CSS with a %c format specifier

for (var i = 0; i < 7; i++) {

    var styles = "font-size: 40px; border-radius: 10px; border: 1px solid blue; background: pink; color: purple";

    console.log("%c" + cubes[i], styles)

}



The second piece of code that's new here is the **cubes[i]** snippet.

This simply targets each individual letter in the loop, based on the current value of the **i** variable.

In other words, **cubes[i]**, when **i** is equal to **0**, is: **A**.

Then, **cubes[i]**, when **i** is equal to **1**, is: **B**.

This goes on for as many loops my for loop runs - and this is determined by the **cubes.length** value.

It's also very versatile, since, if I, for example, decided to change the length of the **cubes** string, I would not have to update the condition of **i < cubes.length**, because it gets automatically updated when I change the length of the **cubes** string.

There are some other ways to store data in JavaScript apps that you haven't heard about.

But we can use the same approach with those other kinds of data, to achieve results that essentially work on the same principle as the one just described.

Using loops is the essence of the approach taken in developing many different pieces of functionality in software today.

**Some additional examples**

If I'm coding an email client, I will get some structured data about the emails to be displayed in the inbox, then I'll use a loop to actually display it in a nicely-formatted way.

If I'm coding an e-commerce site selling cars, I will get a source of nicely-structured data on each of the cars, then loop over that data to display it on the screen.

If I'm coding a calendar online, I'll loop over the data contained in each of the days to display a nicely-formatted calendar.

There are many, many other examples of using loops in code.

Using loops with data that is properly formatted for a given task is a crucial component of building software.

In the lessons that follow, we'll learn about different ways of grouping related data and of displaying it on the screen using JavaScript.

When combined with what you've already learned about loops, this gives you the skills to build various kinds of user interfaces where there is repetitive information.

Some more specific examples include:

* looping over blog post titles in some structured data, and displaying each blog post title on a blog home page
* looping over social media posts in some structured data, and displaying each social media post based on some conditions
* looping over some structured data on clothing available for sale in an online clothing store, and displaying relevant data for each item of clothing

Now you understand the importance of knowing how to work with loops in JavaScript. In the upcoming lessons, we'll learn other relevant information which will allow you to be able to do this.