Coding Basics badge

Song Functions

Learn about functions.

Music is often analyzed from a computer science point of view. Pop songs, kids' songs, and poetry all share structures and patterns that make good examples of algorithms (specific sets of steps to do things). Functions are a common type of instruction in programming that tell a computer to perform a certain task. For example, look at these two functions:

```
    singChorus();         whistle();
```

When you use a function, you would say you're "calling a function," so calling singChorus() will tell the computer to sing the chorus of a song. Calling whistle(); will tell the computer to whistle.

In order for a function to be called, it must be previously defined, or declared. That means that once you have created a function, or defined, a function, you can use it, or call it, again and again in your code. The rules for writing code are called *syntax*.

For example, to call a function with Javascript, you would write it like this:

```
    theNameOfTheFunction();
```

Here are the rules for calling (or writing) a function in JavaScript:

- A function starts with a name, which can't have spaces in it, or use other special characters (except underscores '_').
- The name can include numbers — as long as they're not at the beginning.
- The name is directly followed by parentheses '()'. The parentheses tell JavaScript to run the function.
- A semicolon shows that the function has ended, just as a period ends a sentence in English. A semicolon looks like this: ';'

Here's an example of a function call that is correctly written:

```
    say_promise();
```

But none of these would work:

```
    1st_task();

    make cake;

    do-good ();

    makeCake()again;
```

Why wouldn't they work?

- 1st_eye(); starts with a number.
- make cake; has a space and no parentheses.
- do-good (); uses a hyphen (or minus sign), which JavaScript doesn't allow, and a space between the name and the ().
- makeCake()again; has text after the parentheses.

Learn about arguments.

You can add **arguments** to make your functions more specific. For example:

```
sing();          could sing, but
sing("This Land Is Your Land");      could sing a specific song, and
sing("This Land Is Your Land", name);     could sing a specific song to a specific person
```

By adding details inside the function, you're using arguments to make it more specific. An argument adds details to the function that are changeable, so the sing() function can then be reused in a number of different ways.

In JavaScript, as with many computer languages, arguments are represented as a list separated by commas inside the parentheses. The order of the arguments is important, because the computer will read and execute them in the order, or **sequence,** they're written.

Learn about variables.

Variables hold or "store" information. This makes it easy to reuse information that might be used many times. For example:

```
var person = “Juliette”;
waveTo(person);
greet(person);
```

This code is an example of a variable. It gives the computer instructions to do two things: wave to and greet Juliette. If you want to change the person the computer is greeting and waving to, you only need to make a change in one place instead of multiple places in our program. Saving information in variables is a way to make programs more flexible.

In Javascript, the syntax for using a variable follows this form:

```
var person = theNameOfTheVariable = "this is the value";
```

The variable starts with a name, which can't have spaces in it, or use other special characters (except underscores '_'). The name can use numbers but not at the beginning.

Use functions to write pseudocode for a song.

Fill in the song lyric with a song of your choice on the left side. Then, write the Javascript-style code on the right side.

Use the function sing() to write the code. You might use the same sing function many times but change the arguments for different lyrics. The first argument inside the parentheses is a word or phrase, which can have spaces and punctuation because it's kept inside quotation marks.

For example: sing("This Land Is Your Land") would tell the computer to sing, "This Land Is Your Land."

If needed, checked out the two sample songs, starting on the next page.

Song Lyrics for	Javascript-style code

Coding Basics badge

Example 1- "Make New Friends" (2 verses)

Song Lyrics for	Javascript-style code

Make new friends,
but keep the old.
One is silver,
the other is gold.
 	
A circle is round,
it has no end.
That's how long,
I will be your friend.
Make new friends,
but keep the old.
One is silver,
the other is gold.
 	
A circle is round,
it has no end.
That's how long,
I will be your friend.
	`var verse1 = "Make new friends,`
	`but keep the old.`
	`One is silver,`
	`the other is gold"`
	
	`var verse2 = "A circle is round,`
	`it has no end.`
	`That's how long,`
	`I will be your friend."`
	
	`sing(verse1)`
	`sing(verse2)`
	`sing(verse1)`
	`sing(verse2)`
	
	`//notice that the program is shorter than the original song lyrics when we use functions and variables!`

Example 2 - “We Shall Overcome”

Song Lyrics for	Javascript-style code

We shall overcome
We shall overcome
We shall overcome, some day
	`var verse1 = “We shall overcome”`
	`var verse2 = “We’ll walk hand in hand”`
	`var verse3 = “We shall live in peace”`
	`var verse4 = “We are not afraid”`

Oh, deep in my heart
I do believe
We shall overcome, some day
	`var chorus = “Oh, deep in my heart`
	`                        I do believe`
	`        We shall overcome, some day”`

We’ll walk hand in hand
We’ll walk hand in hand
We’ll walk hand in hand, some day
	`for 1...3`
	`    sing(verse1) // this is repeated 3 times`
	`sing(“some day” + chorus) // no repeat (no indent)`

Oh, deep in my heart
I do believe
We shall overcome, some day
	`for 1...3`
	`    sing(verse2)`
	`sing(“some day”+ chorus)`

We shall live in peace
We shall live in peace
We shall live in peace, some day
	`for 1..3`
	`    sing(verse3)`
	`sing(“some day”+ chorus)`

Oh, deep in my heart
I do believe
We shall overcome, some day
	`for 1..3`
	`    sing(verse4)`
	`sing(“TODAY”)`

We are not afraid We are not afraid We are not afraid, TODAY	`// With loops, you can make things shorter, and, if you wanted to change the verse lyric, you only need to change one line!`