<https://frontendmasters.github.io/bootcamp/functions-and-scope>

Functions and Scope

A function is a bit of re-usable code. Just how we like to re-use CSS classes, we love to re-use code. Let's start with an example:



1

function addTwo(number) {

2

 return number + 2;

3

}

4

​

5

const finalAnswer = addTwo(5);

6

console.log(finalAnswer);



1

7

2

undefined

This isn't super useful but hopefully it shows you the mechanics of how a function works. We created a function called addTwo. This function takes in one parameter, number and it returns that number with 2 added to it. We can now use that addTwo function as much as we want! Let's make a something a bit more useful.



1

function greet(firstName, lastName, honorific, greeting) {

2

 return `${greeting} ${honorific} ${lastName}! I’m extremely pleased you could join us, ${firstName}! I hope you enjoy your stay, ${honorific} ${lastName}.`;

3

}

4

​

5

console.log(greet("Brian", "Holt", "Lord", "Salutations"));

6

console.log(greet("Jack", "Sparrow", "Captain", "A-hoy"));



1

"Salutations Lord Holt! I’m extremely pleased you could join us, Brian! I hope you enjoy your stay, Lord Holt."

2

"A-hoy Captain Sparrow! I’m extremely pleased you could join us, Jack! I hope you enjoy your stay, Captain Sparrow."

3

undefined

Now we rather than have to repeate ourselves over-and-over again with that long string, we can just call greet with the appropriate parameters. Here we use four parameters. The order is important that we send in the parameters because this will be the order function receives these parameters. You can have as many or as few parameters as you like.

The way **call** a function is you add parens to the end of it, like this: someFunctionName(). If you see parens after a variable name, you instantly know that that's a function. Inside of the parens go the parameters. These variables will be passed to the function that is being called in the order that you put them there. Example:



1

const myHomeCity = "Salt Lake City";

2

const myHomeState = "Utah";

3

const myHomeCountry = "USA";

4

​

5

function logOutYourHome(city, state, country) {

6

 console.log(`You are from ${city}, ${state} ${country}.`);

7

}

8

​

9

logOutYourHome(myHomeCity, myHomeState, myHomeCountry);



1

"You are from Salt Lake City, Utah USA."

2

undefined

Scope

We'll talk about scope multiple times but we'll start off here with it. Every time you call a function, it has its own scope. Other things can't peek into it; it just has its own little workspace for it work with. Once its done, any variable that you haven't explicitly held on to or returned at the end is discarded. For example:



1

function addFive(number) {

2

 const someVariable = "you can't see me outside this function";

3

 return number + 5;

4

}

5

​

6

addFive(10);

7

console.log(someVariable);



1

ReferenceError: someVariable is not defined

This is not going to work. someVariable is inside of the addFive scope and once addFive completes, it throws someVariable away since it's now out-of-scope.



1

let friendsAtYourParty = 0;

2

for (let i = 0; i <= 10; i++) {

3

 friendsAtYourParty++;

4

}

5

console.log(i);



1

ReferenceError: i is not defined

Even this doesn't work since i is only in scope for the loop and then after that it's thrown away. This can be a difficult one to deal with as someone new to coding because you'll go to log something or use a variable and it's out of scope so it's not there. Just know if that happens, this is probably the problem.

Scope is hard. And scope is particularly strange in JavaScript (it varies by programming language.) If it feels hard it's because it is. A general, imperfect way for you to think about it right now is that a variable is "alive" (in scope) in between whatever the closest { is until that { closes its corresponding }. A few examples below, see if you can get it right. Keep in mind that the variable will stay in scope as long as any scope it exists in still exists. If I declare a variable in an outter scope and modify a variable in an inner scope, that variable will survive as long as the outter scope does. **It matters where the variable is declared.**

const A = "A";

let F;

function doStuff(B) {

console.log(B);

const C = "C";

let H = "H";

if (1 + 1 === 2) {

const D = "D";

H = "something else";

}

console.log(D);

console.log(H);

F = "F";

}

let E = 0;

while (E < 3) {

E++;

console.log(A);

const G = "G";

}

console.log(E);

console.log(G);

doStuff("B");

console.log(B);

console.log(C);

console.log(F);

This is pretty convulated example but see what you think. Once your ready, the next block will be the answers.



1

const A = "A";

2

let F;

3

​

4

function doStuff(B) {

5

 console.log(B); // works, B parameter is still in scope

6

 const C = "C";

7

 let H = "H";

8

 if (1 + 1 === 2) {

9

   const D = "D";

10

   H = "something else";

11

}

12

 console.log(D); // does not work, D was declared in that if statement block

13

 console.log(H); // works, H was declared outside the if statement

14

 F = "F";

15

}

16

​

17

let E = 0;

18

while (E < 3) {

19

 E++;

20

 console.log(A); // works, the outter block (called the global scope) is still in scope

21

 const G = "G";

22

}

23

console.log(E); // works, E was declared outside the whie loop

24

console.log(G); // does not work, declared inside the while loop and it's over

25

​

26

doStuff("B");

27

console.log(B); // does not work, the B parameter expires after the function call

28

console.log(C); // does not work, C was declared inside the function and the function is over

29

console.log(F); // works, F was declared in the global scope



1

"A"

2

"A"

3

"A"

4

3

5

[Function G]

6

"B"

7

[Function D]

8

"something else"

9

[Function B]

10

ReferenceError: C is not defined

Builtins

Lots of functions already exist for you! Smart people have created this commonly-used functions for things we often need. For example, say you have a string and you want to make everything lowercase, you can do this:



1

const sentence = "ThIs HaS wEiRd CaSiNg On It";

2

console.log(sentence.toLowerCase());



1

"this has weird casing on it"

2

undefined

Always be looking for the parens. And the best place to look all this stuff up is from our friends at Mozilla (makers of Firefox): [the MDN](https://developer.mozilla.org/en-US/). MDN used to stand for "Mozilla Developer Network" I think but now it's just synonmous with the documentation for the web. I literally look at this website several times a day. As I said before, you are not expected to remember everything. Looking things up on the MDN is **not** cheating.

You can call Math.round(5.1) and it'll return that number rounded (in this, 5). You can use string.substr(indexToStart, howManyCharactersToInclude) to return part of a string. For example const name = "Brian Holt"; console.log(name.substr(6, 3)) logs out "Hol" (remember numbering starts at 0). We'll introduce them as we go but know there are a *lot* of them. You'll learn by doing.



1

console.log(Math.round(5.1));

2

​

3

const name = "Brian Holt";

4

console.log(name.substr(6, 3));



1

5

2

"Hol"

3

undefined