Ch 4. Functions

Three common ways to define functions:

1. Function Declarations
   1. *function* name() { [*your code here*] }
2. Function Expressions
   1. Assign an anonymous function to a variable
      1. Ex. *const* goodbye = *function*(){ [*your code here*] }
   2. Or create a named function expression
      1. *const* goodbye = *function* name() { [*your code here*] }
3. Function() Constructors
   1. The body of the function is entered as a string in a constructor
      1. *const* variableName = *new* Function('console.log("Hi World!");');
   2. Not recommended to declare functions this way

IMPORTANT: DRY – “Don’t repeat Yourself”, the principle of programming that specifies that every part of a program should only be written once.

Invoking a function requires parentheses. Typing the name without them references the function itself. The latter is useful for assigning the function to another variable.

All functions return a value. If it doesn’t explicitly return anything, it will return **undefined** by default.

Functions have a special variable called **arguments**. It’s an array-like object that contains every argument passed to the function when it is invoked. A function that returns **arguments** allows access to its values with index notation:

*function* arguments() {

*return* arguments;

}

Access values with index notation (arguments[0]).

**rest operator**

A **rest ...** operator can be used to deal with multiple arguments by creating an array of arguments that are available inside the body of the function:

*function* rest(...args){

*return* args;

}

The **args** parameter is the actual array. A *for-of* loop can then access and iterate over each value given as an argument:

*function* rest(...args){

*for*(arg *of* args){

console.log(arg);

}

}

rest(2,4,6,8);

<< 2

4

6

8

**Default Parameters**

Default parameters are values used by a function if no arguments are provided when it is invoked:

*function* hello(name=’World’) {

console.log(`Hello ${name}!);

}

Obviously, specifying an argument will override the default value. Default parameters should always come after non-default parameters [*function* discount(price, amount=10)].

**Arrow Functions**

Declaring functions can be done with an **arrow =>**. Parameters come before the arrow, with the main body of the function coming after. Arrow functions are **always anonymous**, and must be assigned to a variable:

*const* square = x => x\*x;

Advantages of arrow functions:

* less verbose than other declarations
* single parameters don’t need parentheses
* function body doesn’t need to be placed into a block if it’s a single line
* **return** keyword isn’t required if return statement is the only statement in the body of the function
* Don’t bind their own value of **this** to the function.

Multiple parameters still need to be in parentheses. No parameters? Need empty parentheses.

Longer functions require curly braces and a return keyword at the end.

Remember to declare functions at the top of your code, and variables at the top of each block when possible.

**Callbacks**

a function that is passed as an argument to another is known as a *callback*:

*function* sing(song,callback) {

console.log(`I’m singing along to ${song}.`);

callback();

}

Any function can be used as a callback. When passed as an argument, the callback is written without parentheses. It is invoked in the body of the function.