W02 - Functions
ES6 introduced "arrow" notation

Named functionwith function declaration:

```
function hello(){ Console.log('Hello World!'); }
```

Function EXPRESSION - assigns an anonymous function to variabl
- or named function

```
Const goodbye = function(){ console.log('Goodbye, world!'); };
Const goodbye = function bye(){ console.log('Goodbye!'); };
```

- Semicolon at end of assignment

Function Constructors: new Function(); not recommended

```
const hi = new Function('console.log("Hi!");');
```

THIS CAN CAUSE PROBLEMS. ALWAY GLOBAL.

DRY code: "do not repeat yourself"

- ALL FUNCTION RETURN A VALUE → unexplicite: undefined

Parameters → defined when function is written
Arguments → are values used as those parameters

rest (...) operator

```
function rest(... args){ do something }
```

Collects all arguments entered into an array
can use array methods to access the arguments

DEFAULT PARAMETERS -
```
function hello(name='world'){
    console.log('Hello ${name}!');
}
```

when multiple parameters, default params come last

ARROW FUNCTIONS - alway anonymous functions

```
const add = (x,y) => x + y;
```

multiple parameters use parenthesis. no parameters use empty ()
Longer function still use curly brackets

W02- Functions

FUNCTION HOISTING → All functions are available regardless of where they exist in code

VARIABLE HOISTING → var keyword declared variables are hoisted

Do not rely on hoisting

Functions behave as every other object

CALLBACKS -

```
function sing (song, callback) {
    console.log(`I'm singing along to ${song}.`);
    callback();
  }
```

Callback is function name that is invoked within function
argument is passed without parentheses - it's a reference

```
function numerically (a,b) { return a-b; }

 [1,3,12,5,23,8,7].sort(numerically);

or function numerically (a,b) {      // use to improve .sort()
     if (a < b) { return -1;
     } else if (a > b) { return 1;
     } else { return 0; } }
```

Array Iterators and Callbacks

.forEach(callbackFunction)

```
const colors = ['Red','Green','Blue']
for (let i=0, max = colors.length; i < max; i++) {
   console.log(`Color at position ${i} is ${colors[i]}`);
 }
```

Use for-each to iterate through the array:

```
colors.forEach((color, index) => console.log(`Color at position
     ${index} is ${color}`));
```

callback function

.map(callbackFunction)   `[1,2,3].map(x => 2*x);`

.reduce(callbackFunction) ← utilizes callback function to describe how to combine each value

.filter (callbackFunction) ← callback describes how to filter array

W02 - Functions

CHAINING ITERATORS - combine iterators to increase transform

```
[1,2,3].map(x => x * x).reduce((acc, x) => acc + x);
  (1 * 1) + (2 * 2) + (3 * 3) = 14
```