# **Exercise on Arrow Functions**

You will be writing and modifying function declarations using the arrow syntax.

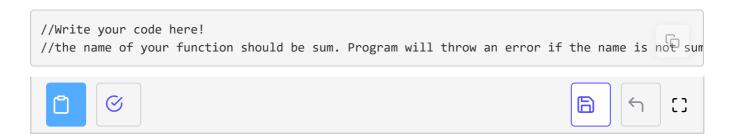
#### Exercise 1:

Write an arrow function that returns the string 'Hello World!'.



### Exercise 2:

Write an arrow function that expects an array of integers, and returns the sum of the elements of the array. Use the built-in method reduce on the array argument.



#### **Explanation:**

Reduce works with an accumulator to store the value associated with reducing the array, and takes two arguments:

- the initial value of the accumulator,
- a function to define the operation between the accumulator and the upcoming element of the array. Reduce performs the following operations:
- (0,1)=>0+1 becomes 1,
- (1,2)=>1+2 becomes 3,
- (3,3)=>3+3 becomes 6,

- (6,4)=>6+4 becomes 10,
- (10,5) = > 10 + 5 becomes 15.

## Exercise 3:

Rewrite the following code by using arrow functions wherever it makes sense to use them:

```
var Entity = function( name, delay ) {
    this.name = name;
    this.delay = delay;
};
Entity.prototype.greet = function() {
    setTimeout(function() {
        console.log( 'Hi, I am ' + this.name );
        }.bind( this ), this.delay );
};

var java = new Entity( 'Java', 5000 );
var cpp = new Entity( 'C++', 30 );
java.greet();
cpp.greet();
```

### **Explanation:**

- It does not make sense to replace the Entity constructor, because we need the context.
- It does not make sense to replace the prototype extension greet, as we make use of its default context.
- It makes perfect sense to replace the function argument of setTimeout with an arrow function. Notice that the context binding also disappeared in the solution.