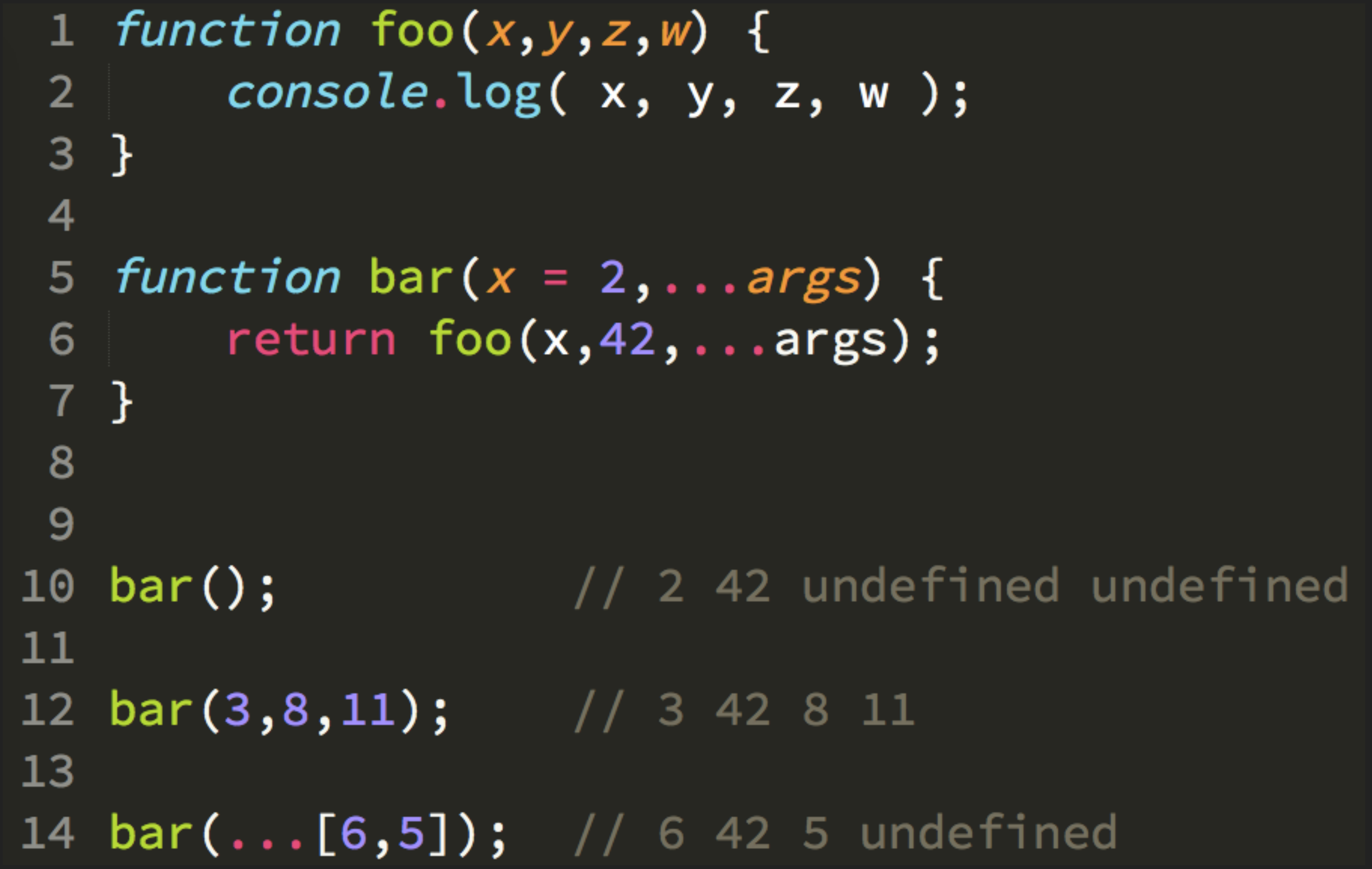
KYLE SIMPSON GETIFY@GMAIL.COM

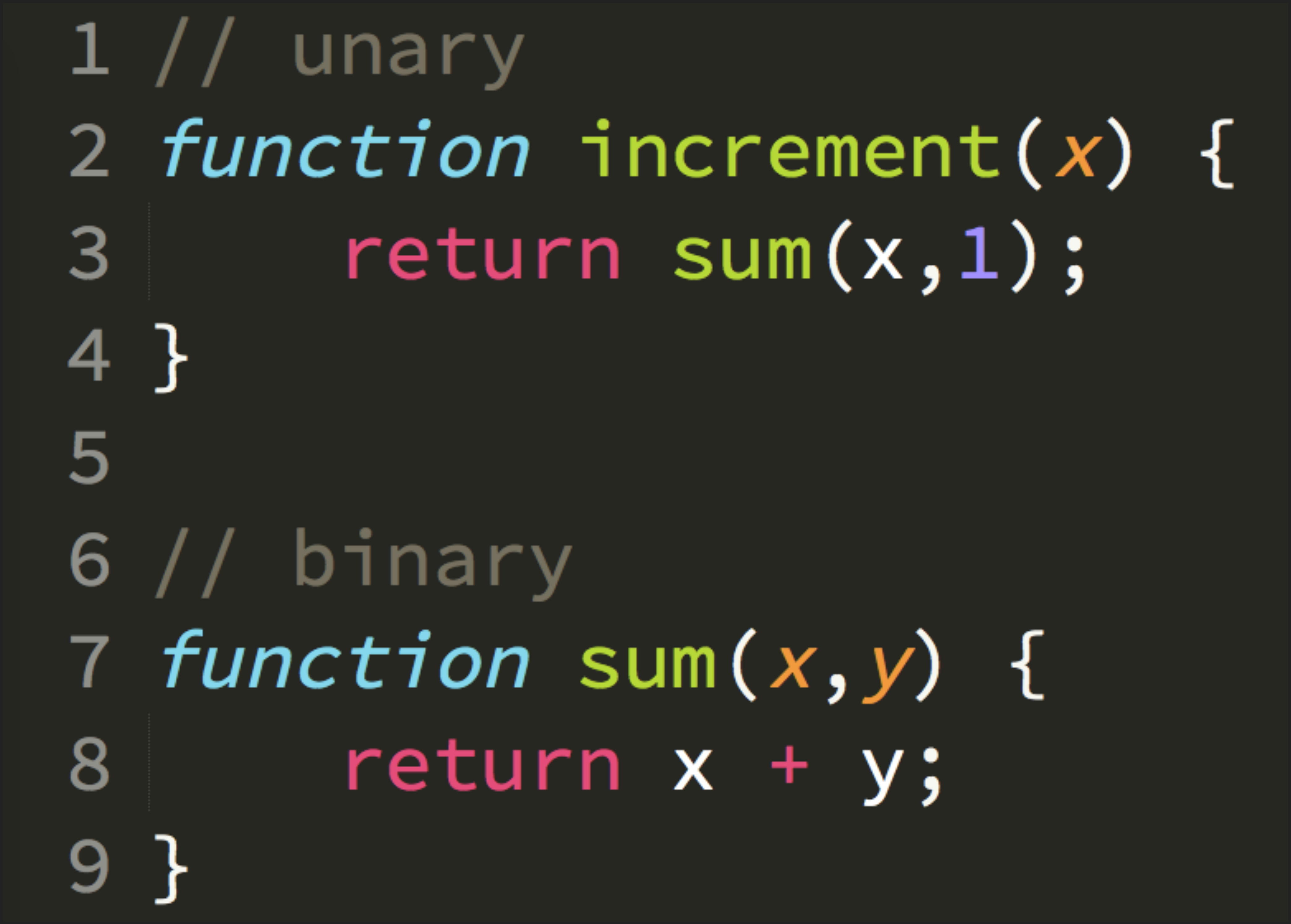
FUNCTIONAL-LIGHT JS

FUNCTIONS

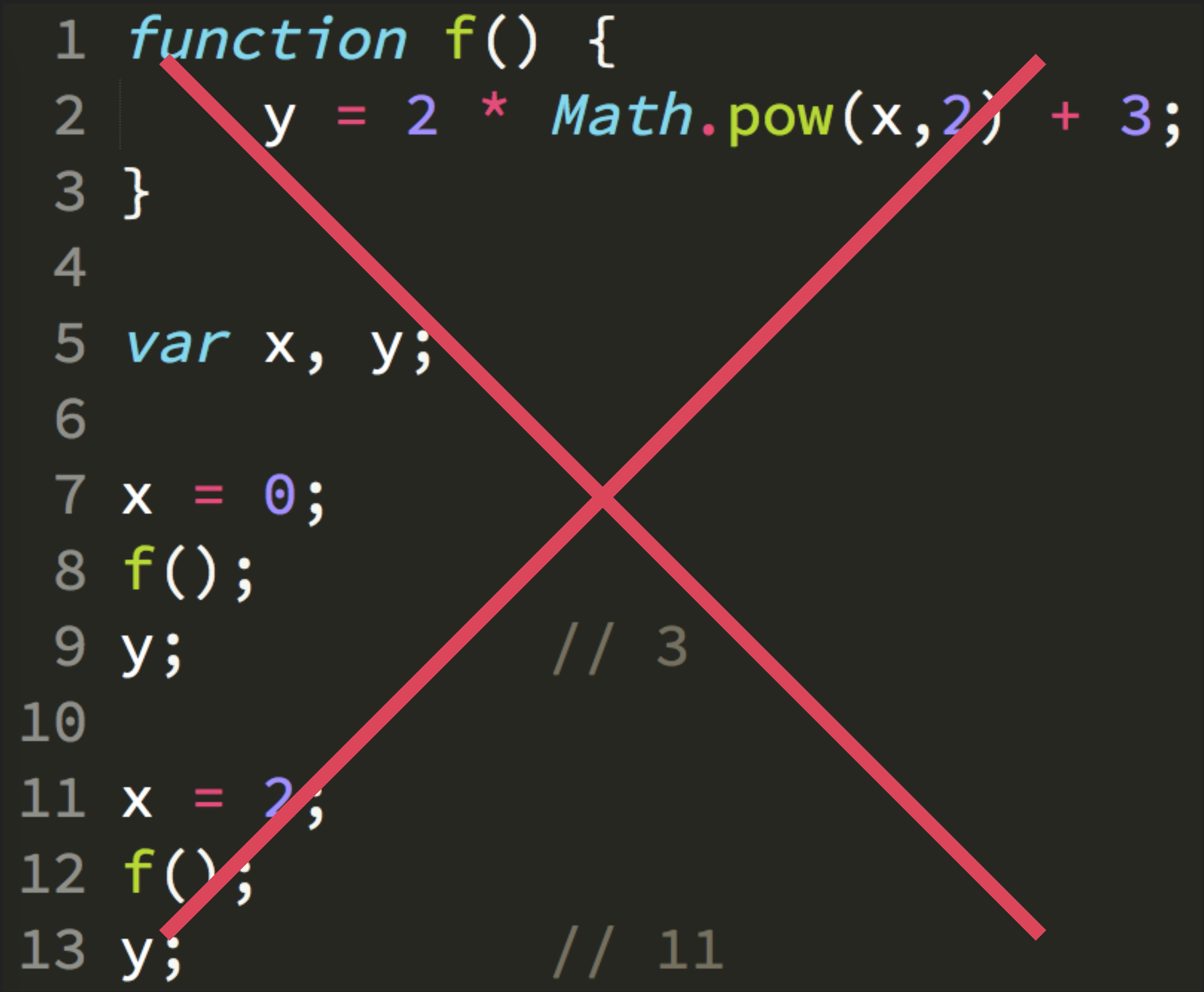


```javascript
function foo(x,y,z,w) {
    console.log( x, y, z, w );
}

function bar(x = 2,...args) {
    return foo(x,42,...args);
}


bar();              // 2 42 undefined undefined

bar(3,8,11);        // 3 42 8 11

bar(...[6,5]);      // 6 42 5 undefined
```

```
function foo(x,y) {
    return [x + 1, y - 1];
}

var [a,b] = foo(...[5,10]);

a;          // 6
b;          // 9
```

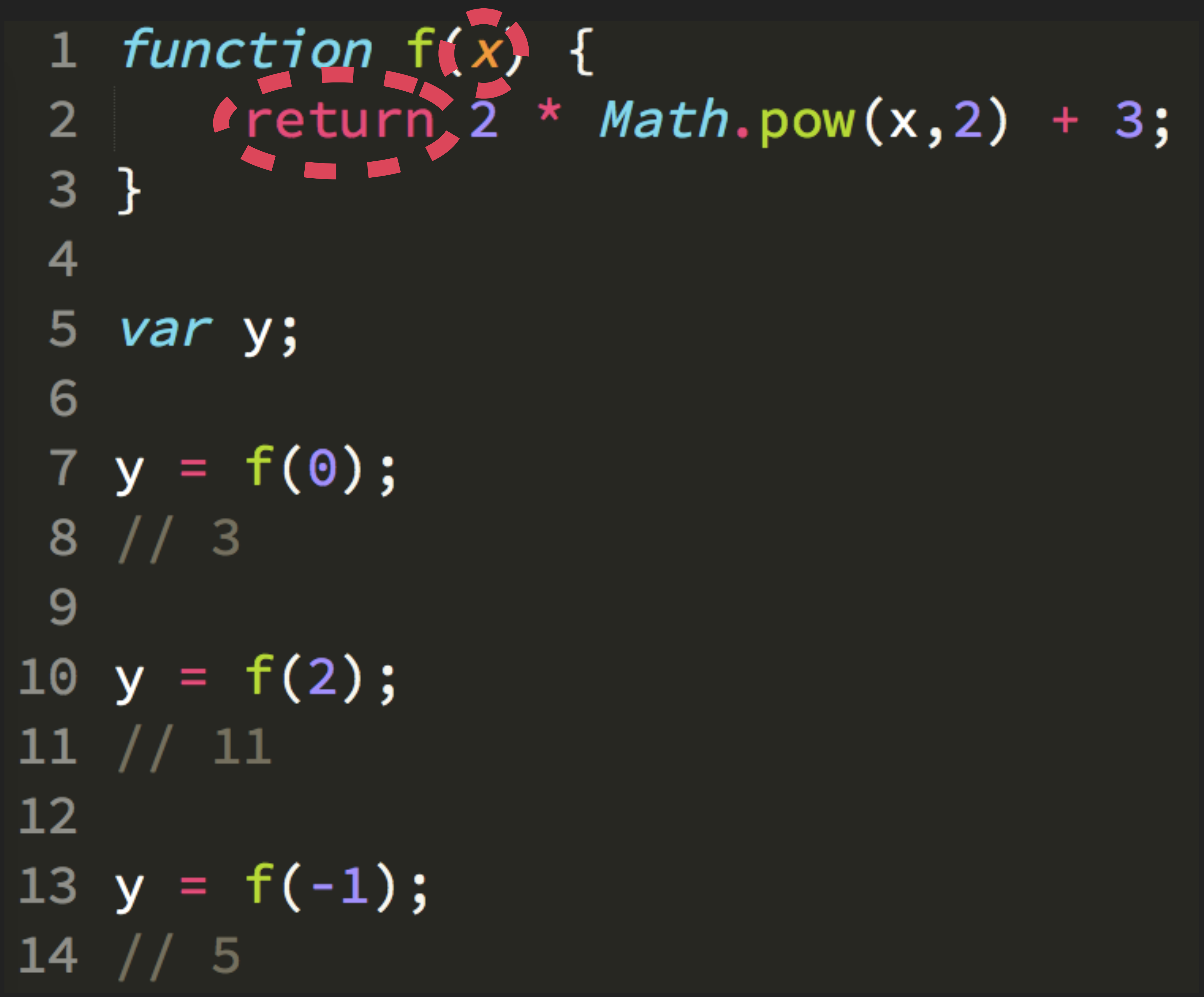


```
// unary
function increment(x) {
    return sum(x,1);
}

// binary
function sum(x,y) {
    return x + y;
}
```



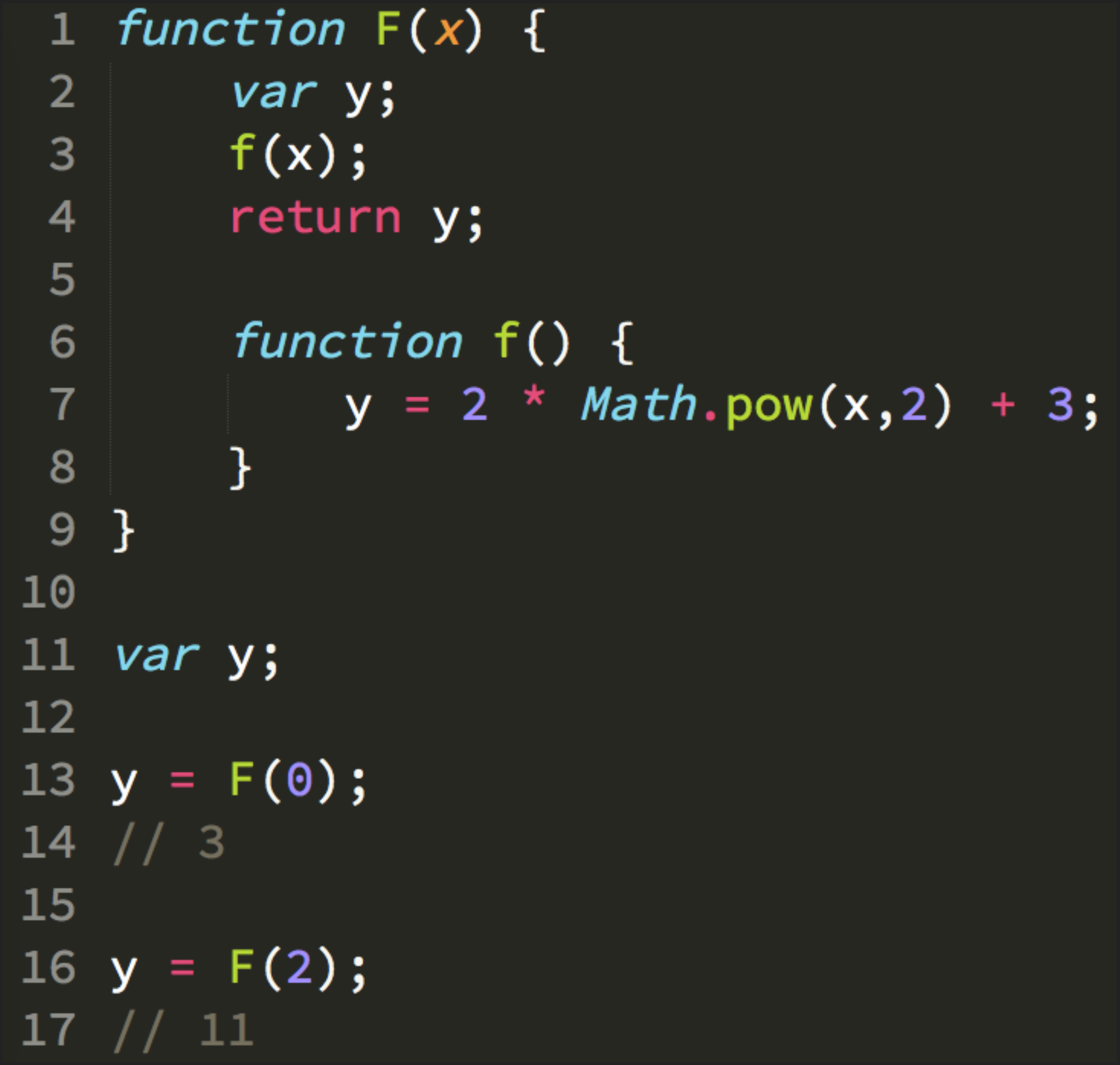
```javascript
function f() {
    y = 2 * Math.pow(x,2) + 3;
}

var x, y;

x = 0;
f();
y;                // 3

x = 2;
f();
y;                // 11
```

~~SIDE EFFECTS~~

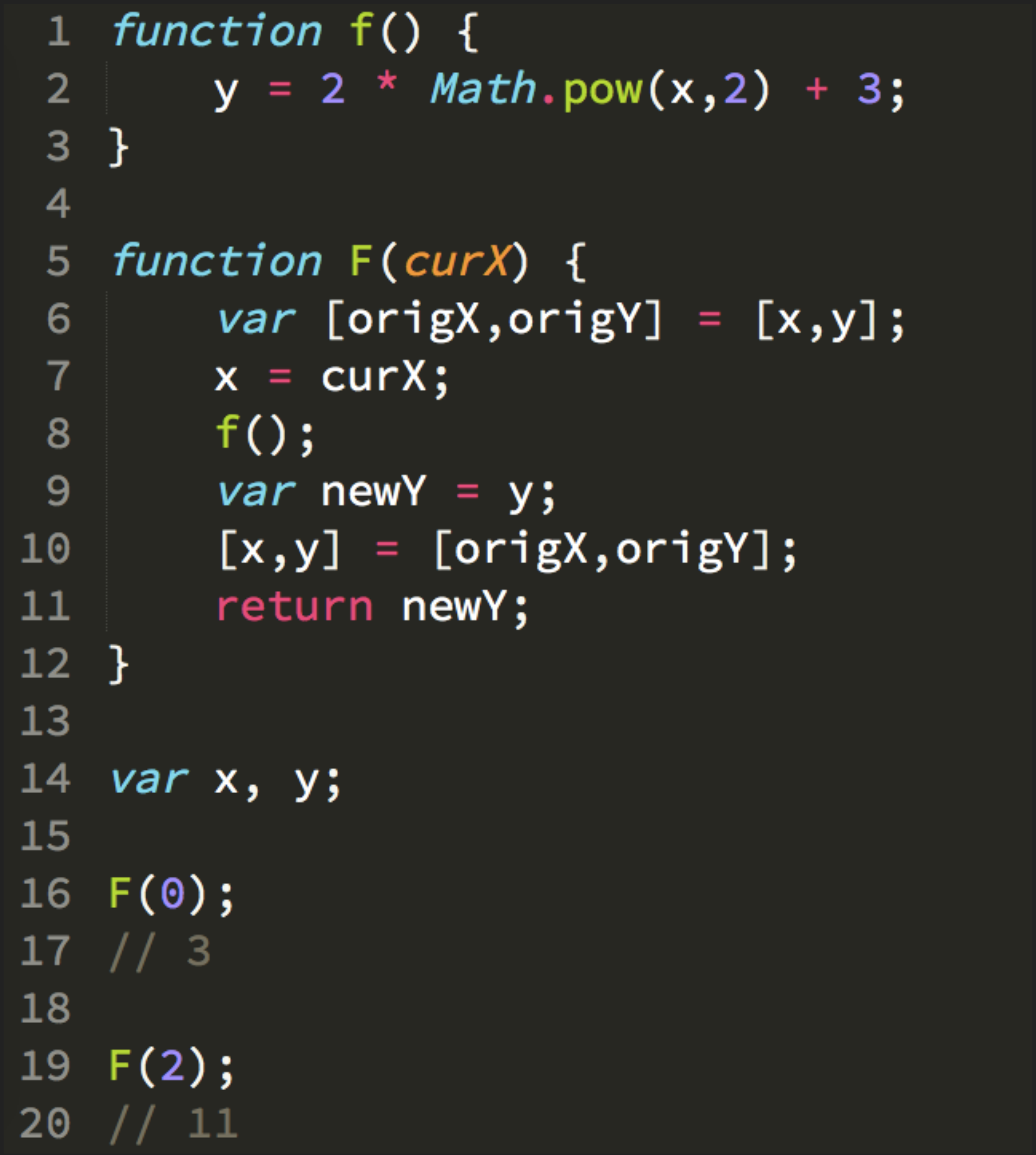


```
function f(x) {
    return 2 * Math.pow(x,2) + 3;
}

var y;

y = f(0);
// 3

y = f(2);
// 11

y = f(-1);
// 5
```

PURE FUNCTIONS



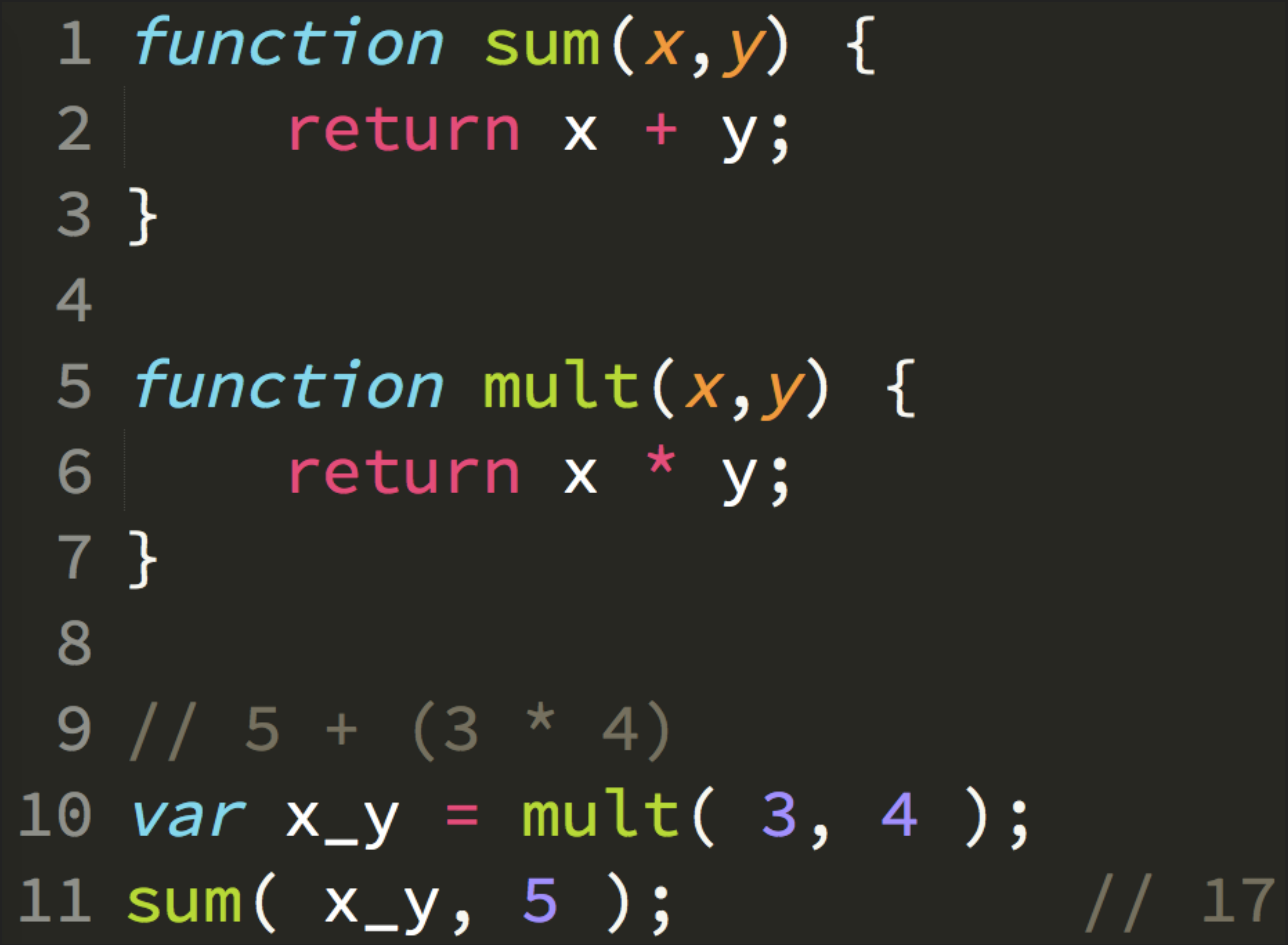
```
function F(x) {
    var y;
    f(x);
    return y;

    function f() {
        y = 2 * Math.pow(x,2) + 3;
    }
}

var y;

y = F(0);
// 3

y = F(2);
// 11
```



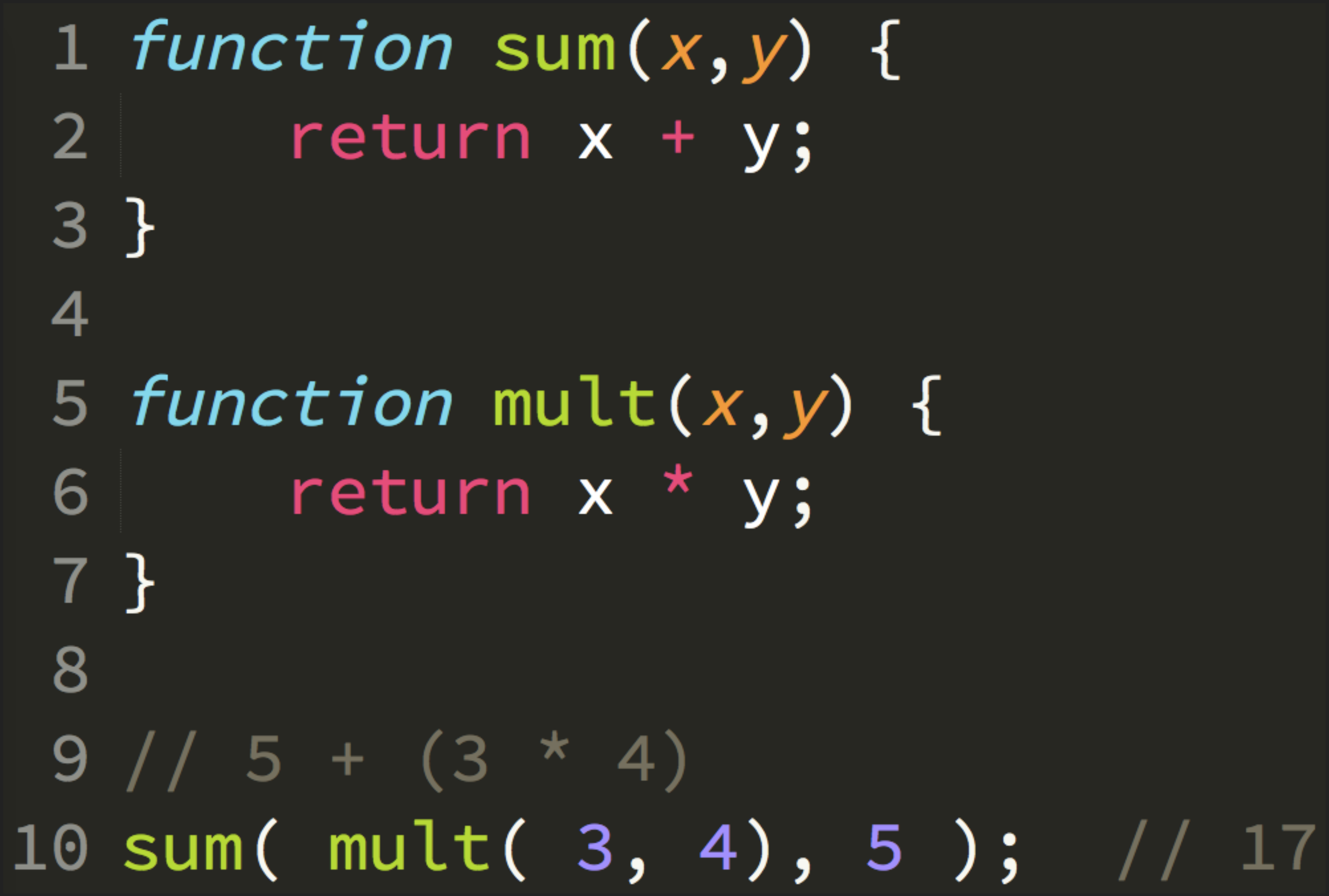
```
function f() {
    y = 2 * Math.pow(x,2) + 3;
}

function F(curX) {
    var [origX,origY] = [x,y];
    x = curX;
    f();
    var newY = y;
    [x,y] = [origX,origY];
    return newY;
}

var x, y;

F(0);
// 3

F(2);
// 11
```

EXERCISE 1

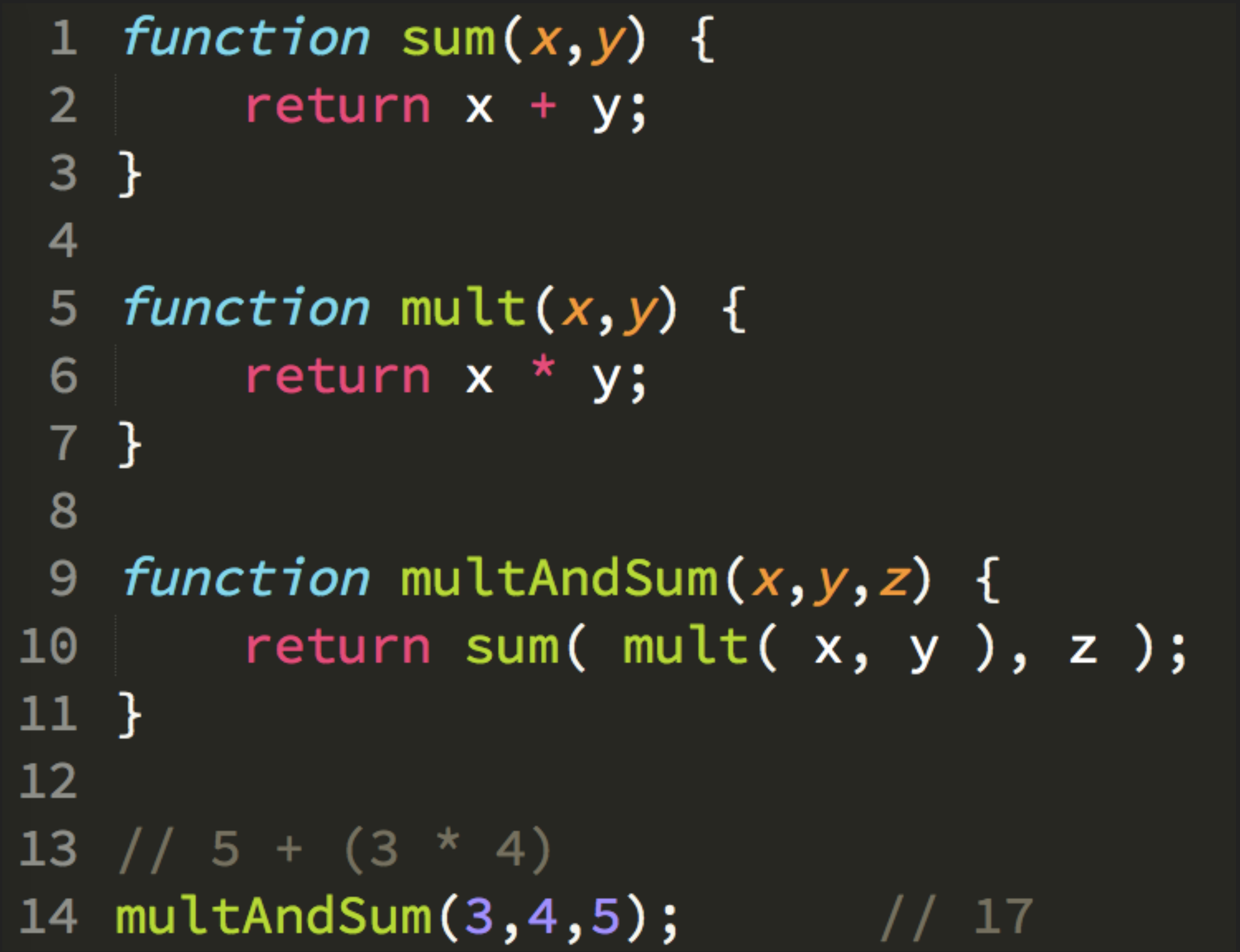
COMPOSITION



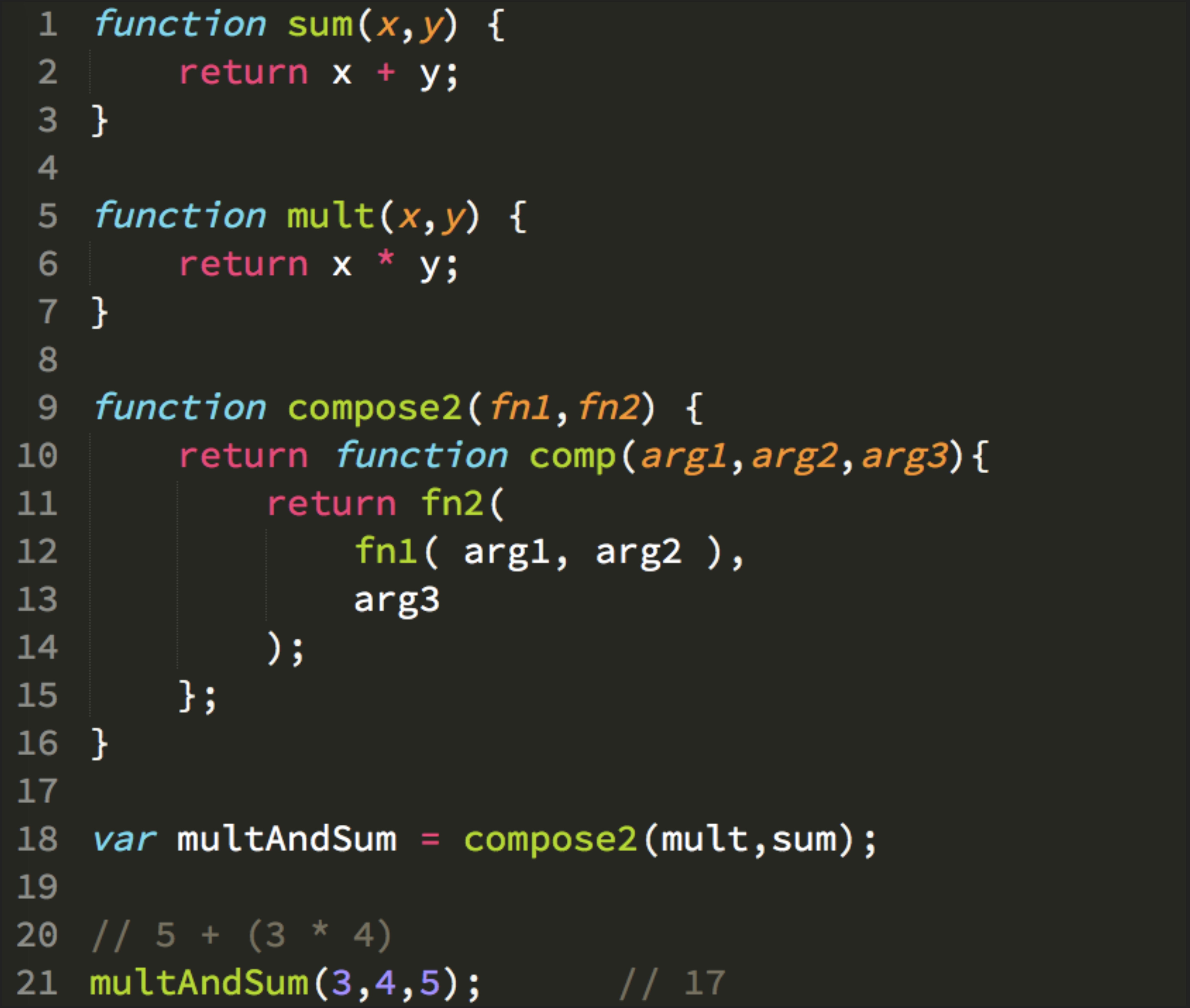
```
function sum(x,y) {
    return x + y;
}

function mult(x,y) {
    return x * y;
}

// 5 + (3 * 4)
var x_y = mult( 3, 4 );
sum( x_y, 5 );                 // 17
```



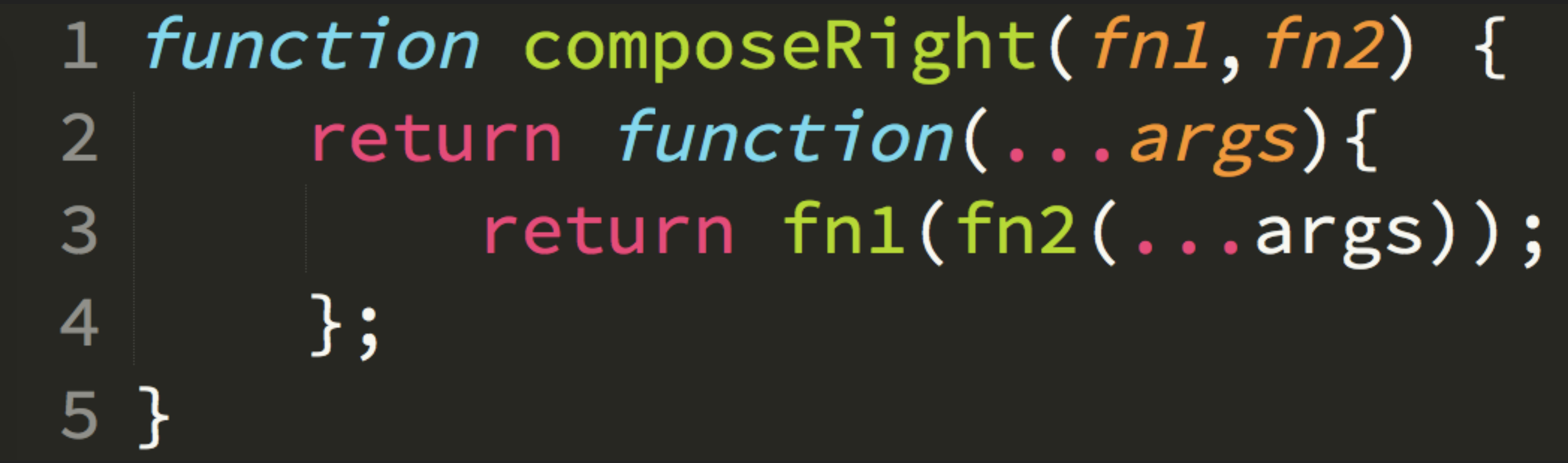
```javascript
function sum(x,y) {
    return x + y;
}

function mult(x,y) {
    return x * y;
}

// 5 + (3 * 4)
sum( mult( 3, 4), 5 );  // 17
```



```
function sum(x,y) {
    return x + y;
}

function mult(x,y) {
    return x * y;
}

function multAndSum(x,y,z) {
    return sum( mult( x, y ), z );
}

// 5 + (3 * 4)
multAndSum(3,4,5);          // 17
```



```javascript
function sum(x,y) {
    return x + y;
}

function mult(x,y) {
    return x * y;
}

function compose2(fn1,fn2) {
    return function comp(arg1,arg2,arg3){
        return fn2(
            fn1( arg1, arg2 ),
            arg3
        );
    };
}

var multAndSum = compose2(mult,sum);

// 5 + (3 * 4)
multAndSum(3,4,5);          // 17
```



```
function composeRight(fn1,fn2) {
    return function(...args){
        return fn1(fn2(...args));
    };
}
```

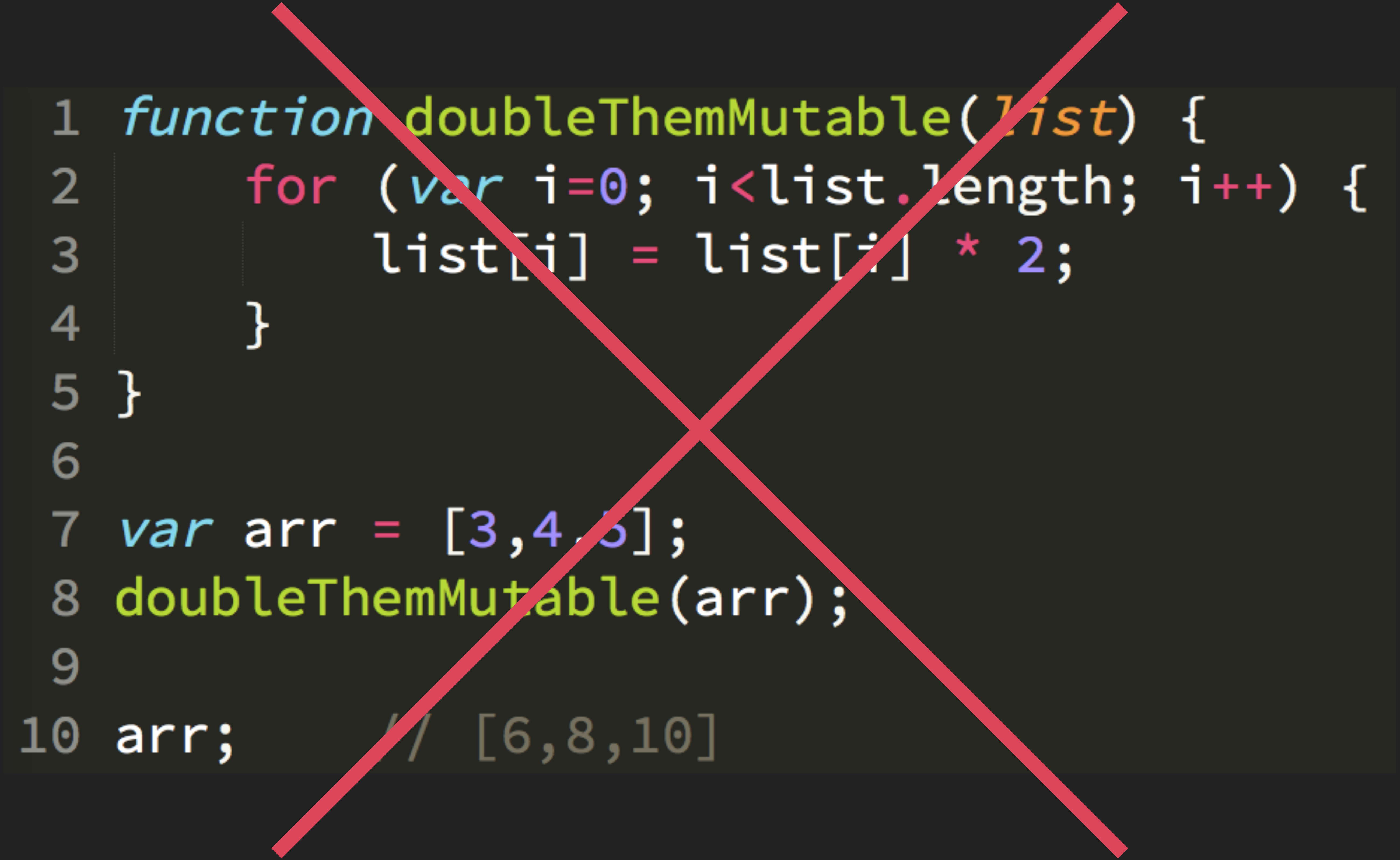
```javascript
function increment(x) {
    return x + 1;
}

function double(x) {
    return x * 2;
}

var f = composeRight(increment,double);
var p = composeRight(double,increment);

f(3);   // 7
p(3);   // 8
```

EXERCISE 2

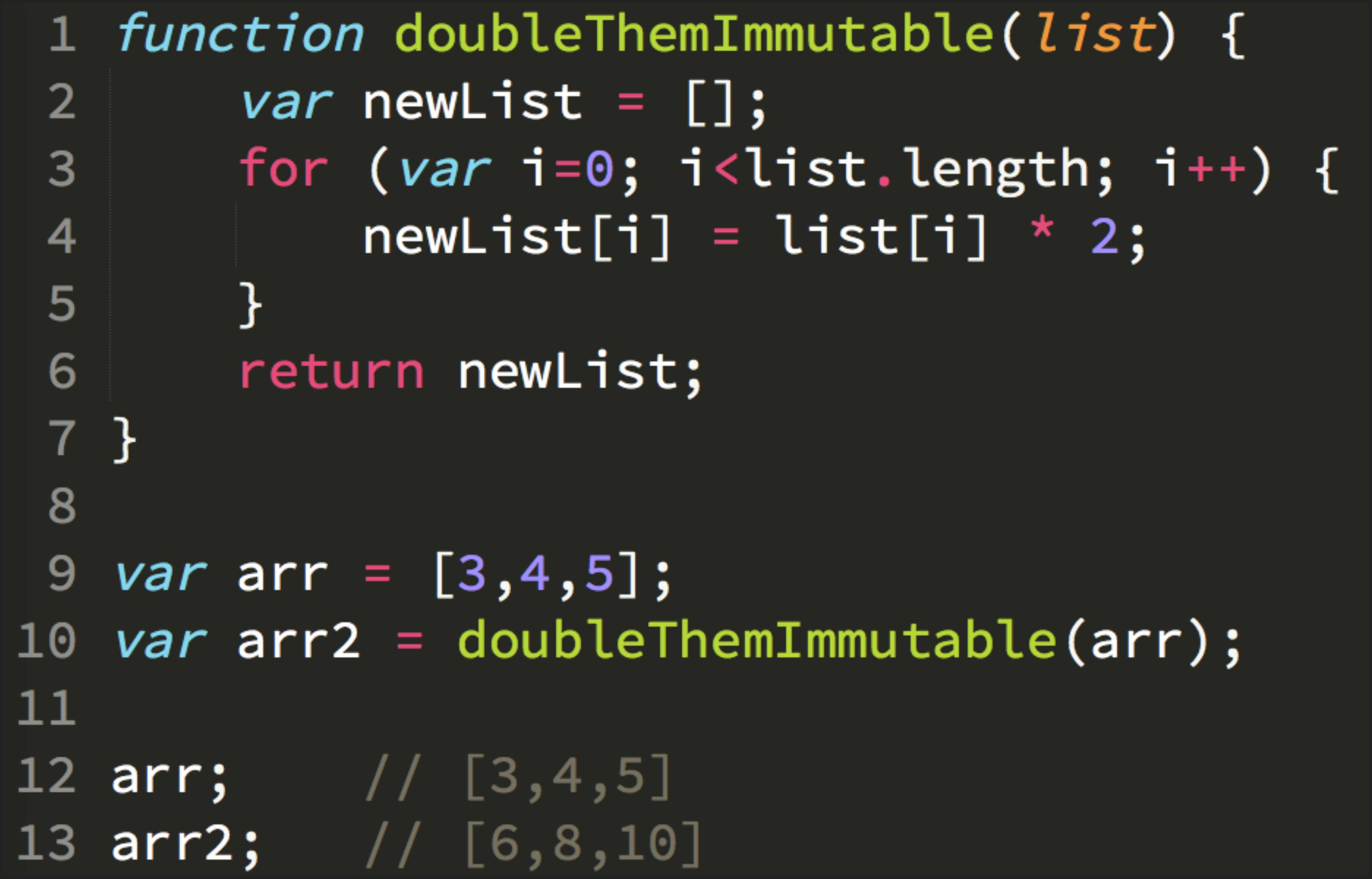
IMMUTABILITY

```javascript
var x = 2;
x++;                // allowed

const y = 3;
y++;                // not allowed

const z = [4,5,6];
z = 10;             // not allowed
z[0] = 10;          // allowed!
```

```
var z = Object.freeze([4,5,6,[7,8,9]]);

z[0] = 10;       // not allowed
z[3][0] = 10;    // allowed!
```



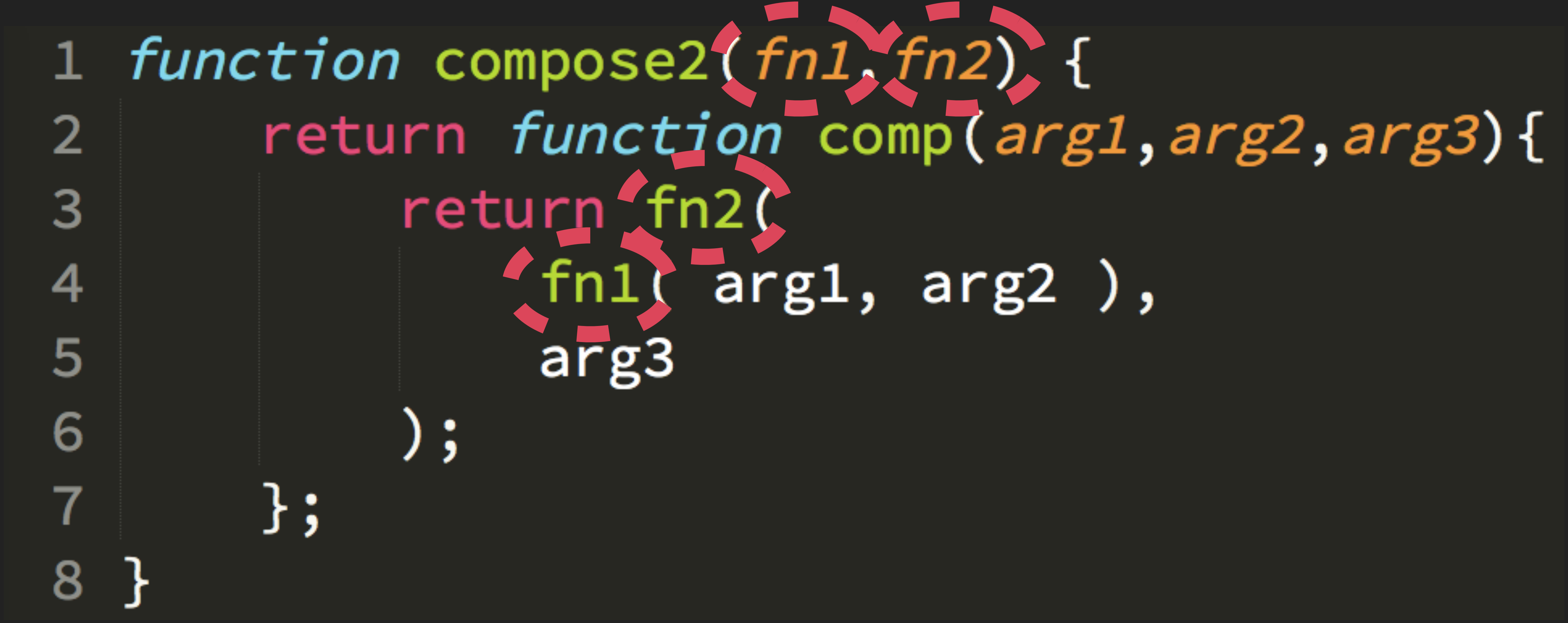
```
function doubleThemMutable(list) {
    for (var i=0; i<list.length; i++) {
        list[i] = list[i] * 2;
    }
}

var arr = [3,4,5];
doubleThemMutable(arr);

arr;      // [6,8,10]
```



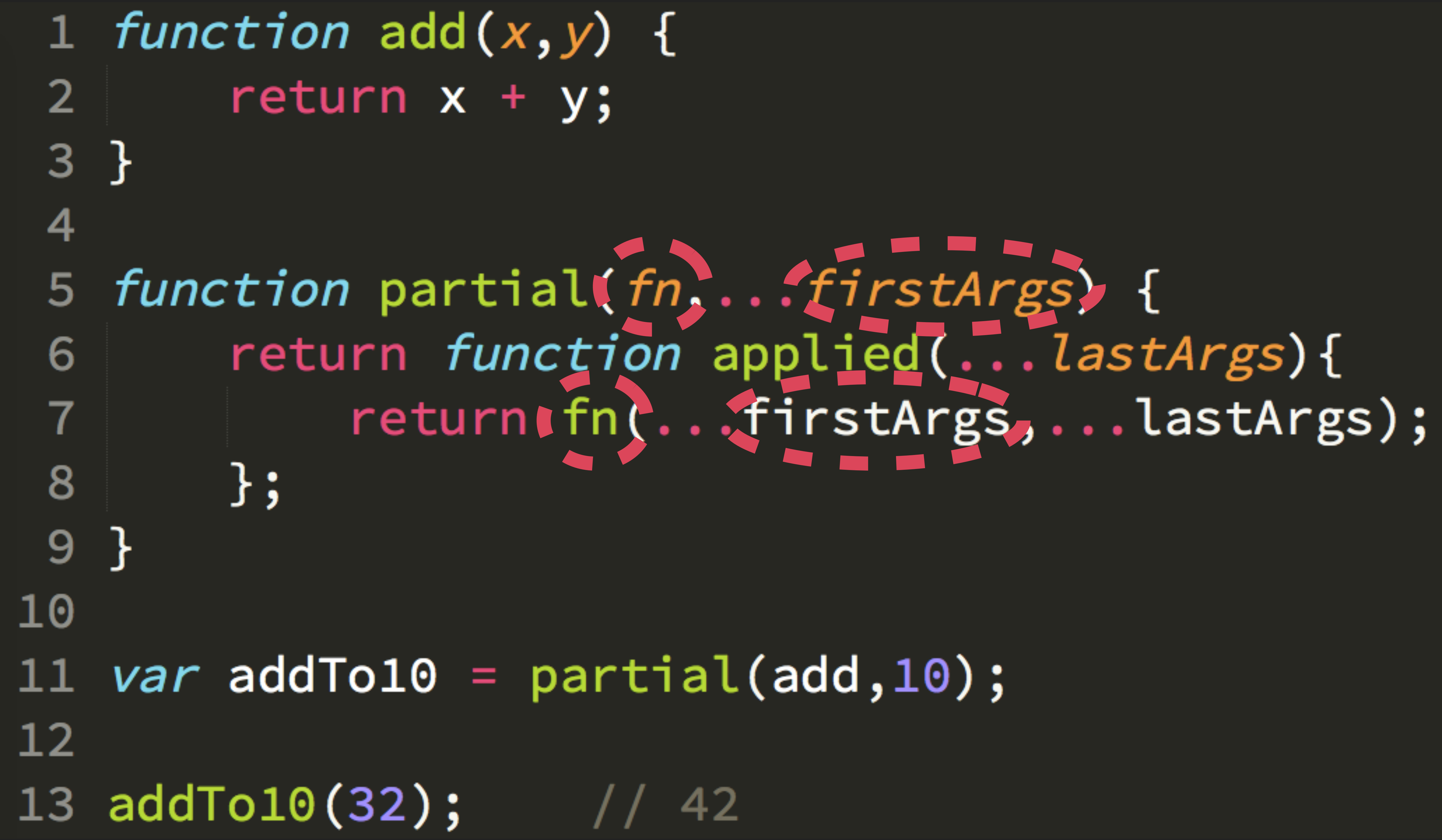
```javascript
function doubleThemImmutable(list) {
    var newList = [];
    for (var i=0; i<list.length; i++) {
        newList[i] = list[i] * 2;
    }
    return newList;
}

var arr = [3,4,5];
var arr2 = doubleThemImmutable(arr);

arr;    // [3,4,5]
arr2;   // [6,8,10]
```

CLOSURE

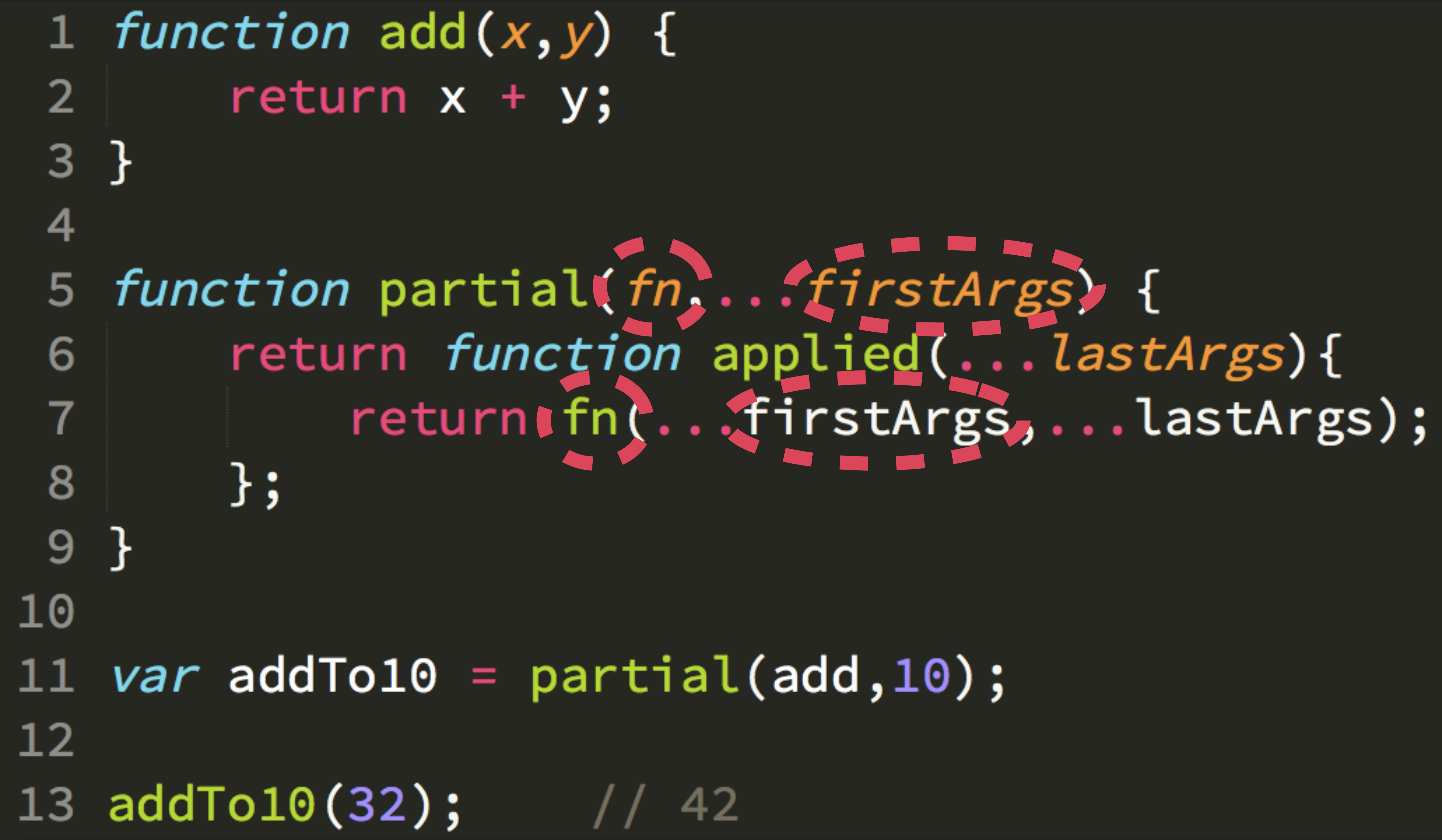
Closure is when a function "remembers" the variables around it even when that function is executed elsewhere.



```
function compose2(fn1,fn2) {
    return function comp(arg1,arg2,arg3){
        return fn2(
            fn1( arg1, arg2 ),
            arg3
        );
    };
}
```



```javascript
function add(x,y) {
    return x + y;
}

function partial(fn,...firstArgs) {
    return function applied(...lastArgs){
        return fn(...firstArgs,...lastArgs);
    };
}

var addTo10 = partial(add,10);

addTo10(32);     // 42
```



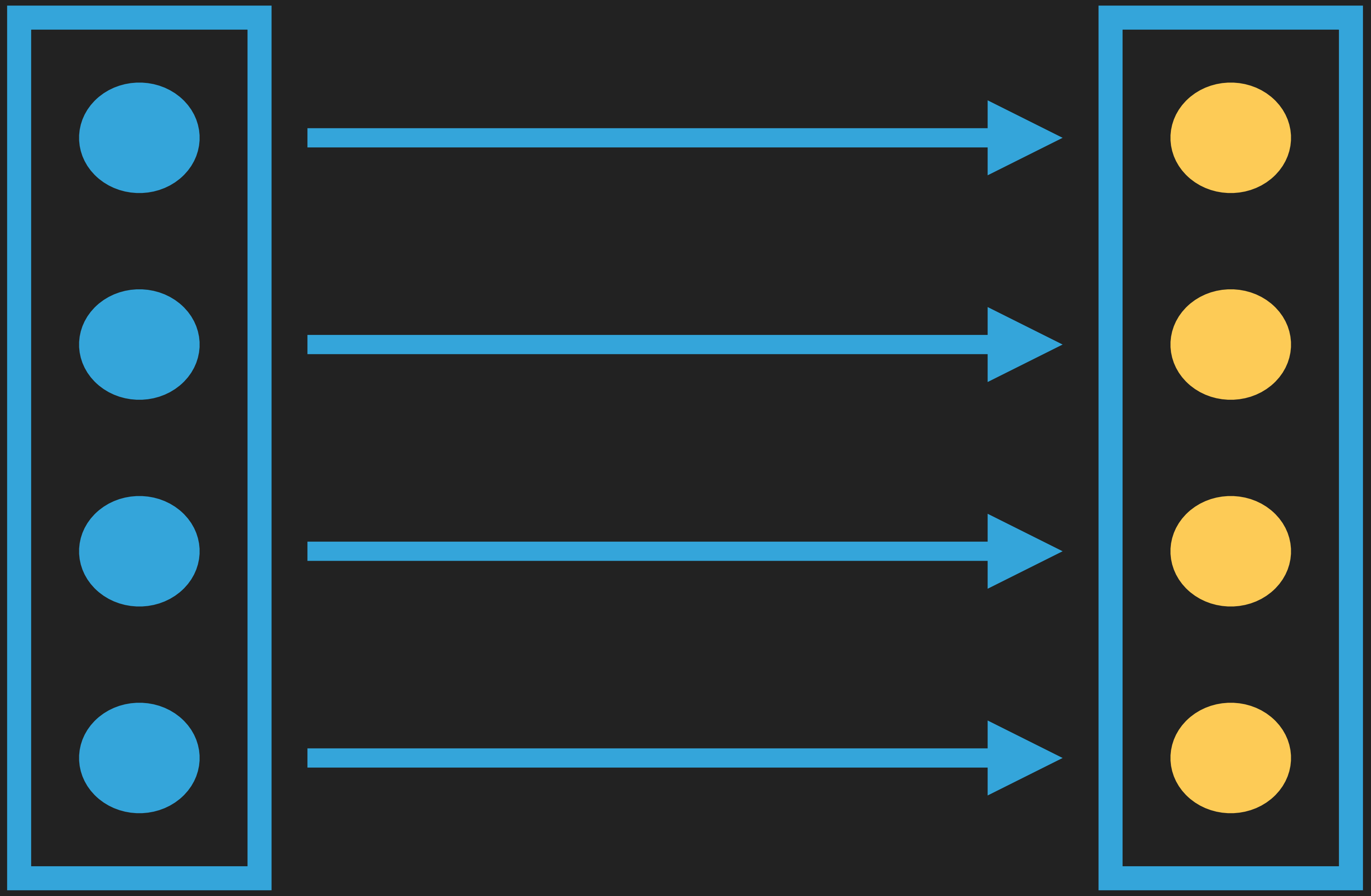
```
function add(x,y) {
    return x + y;
}

function partial(fn,...firstArgs) {
    return function applied(...lastArgs){
        return fn(...firstArgs,...lastArgs);
    };
}

var addTo10 = partial(add,10);

addTo10(32);     // 42
```

```javascript
var add3 = curry(function add3(x,y,z) {
    return x + y + z;
});

var f = add3(3);

var p = f(4);

p(5);             // 12

add3(3)(4)(5);  // 12
```

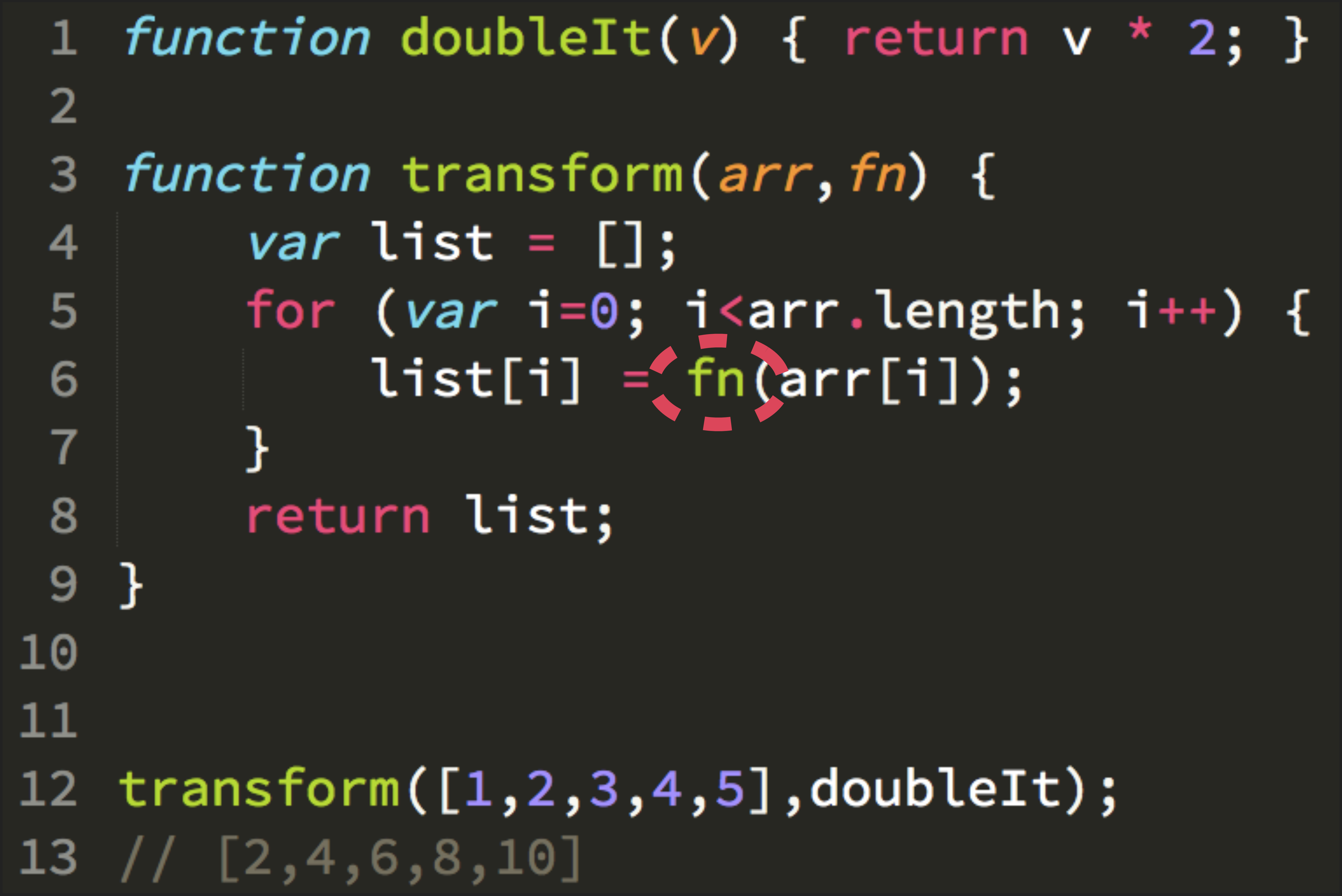
EXERCISE 3

If you can do something awesome,
keep doing it repeatedly.

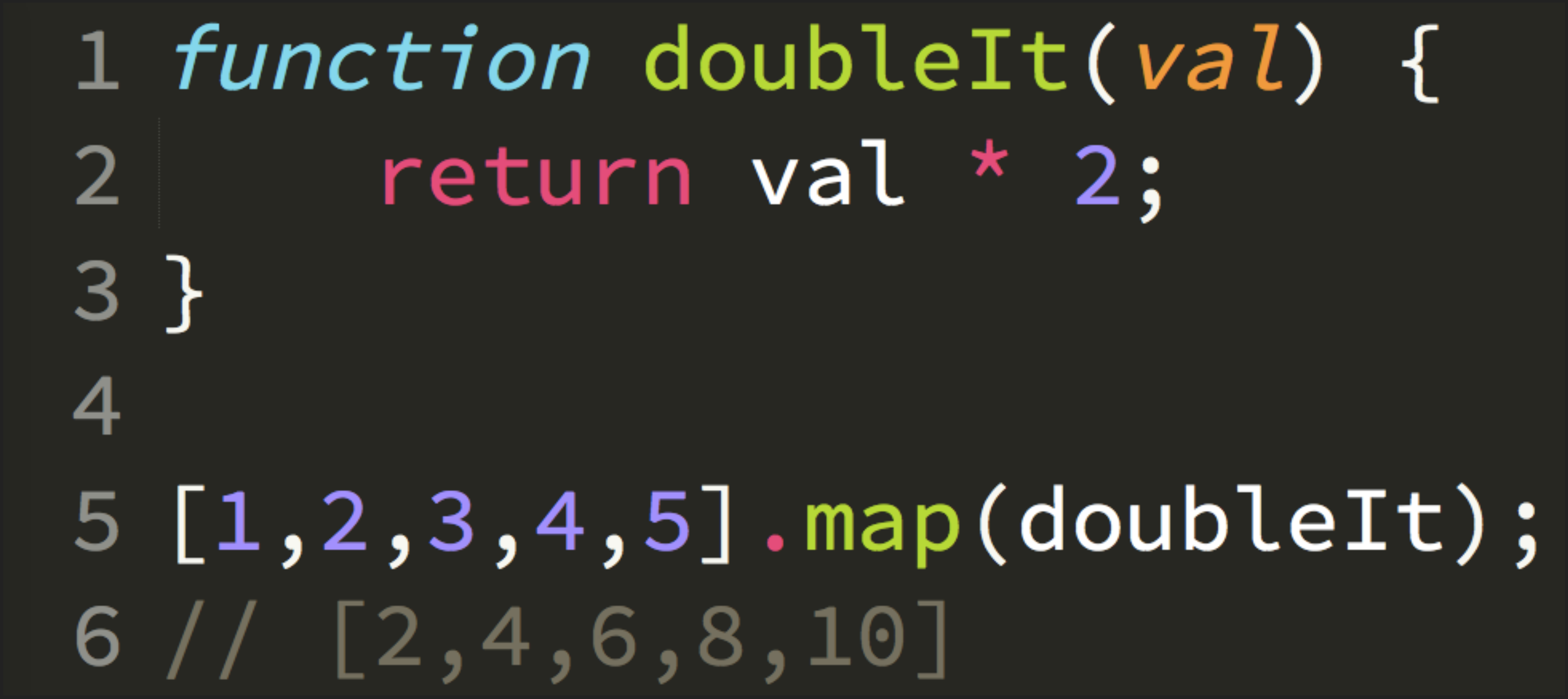
LISTS



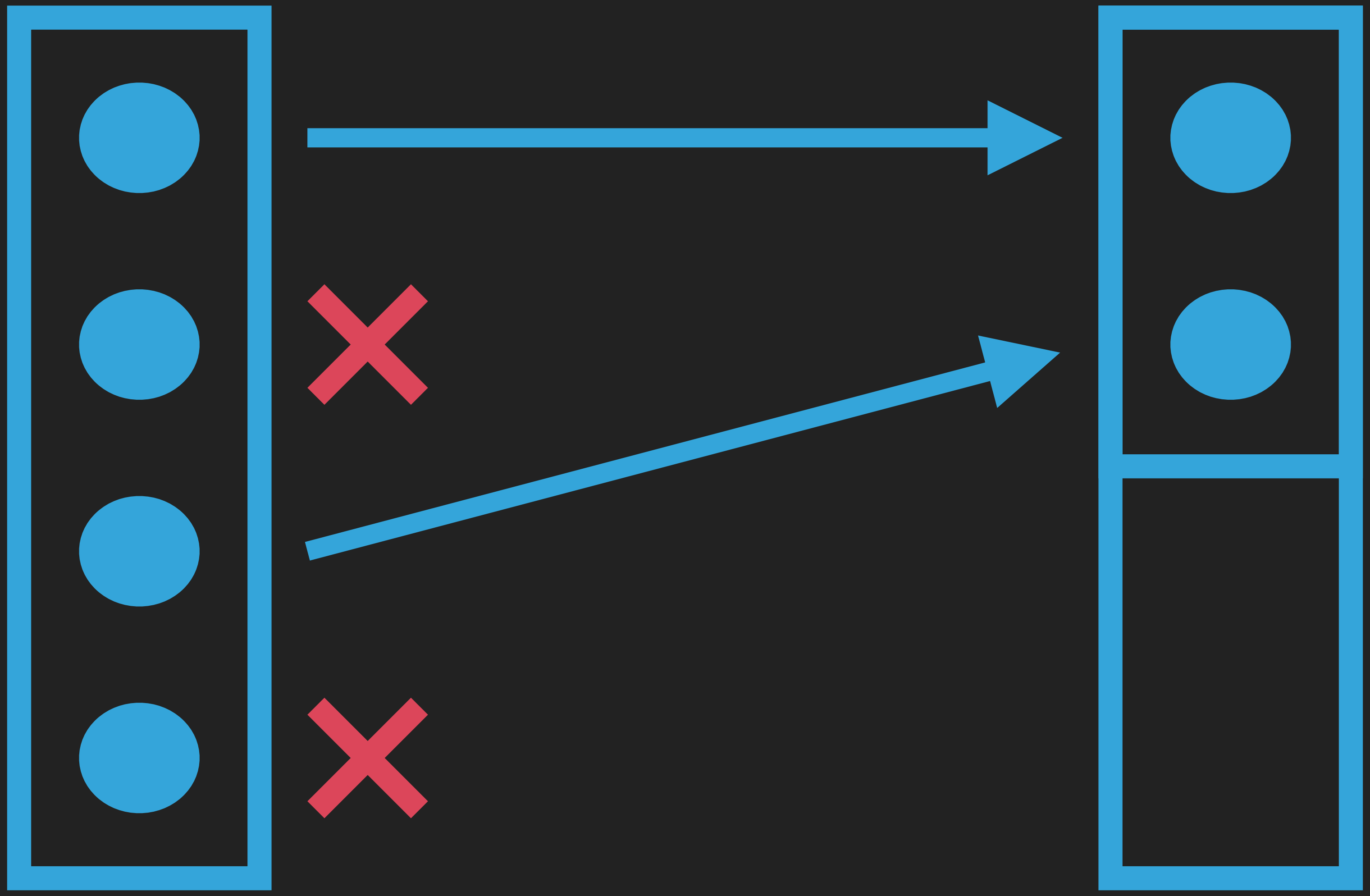
MAP: TRANSFORMATION



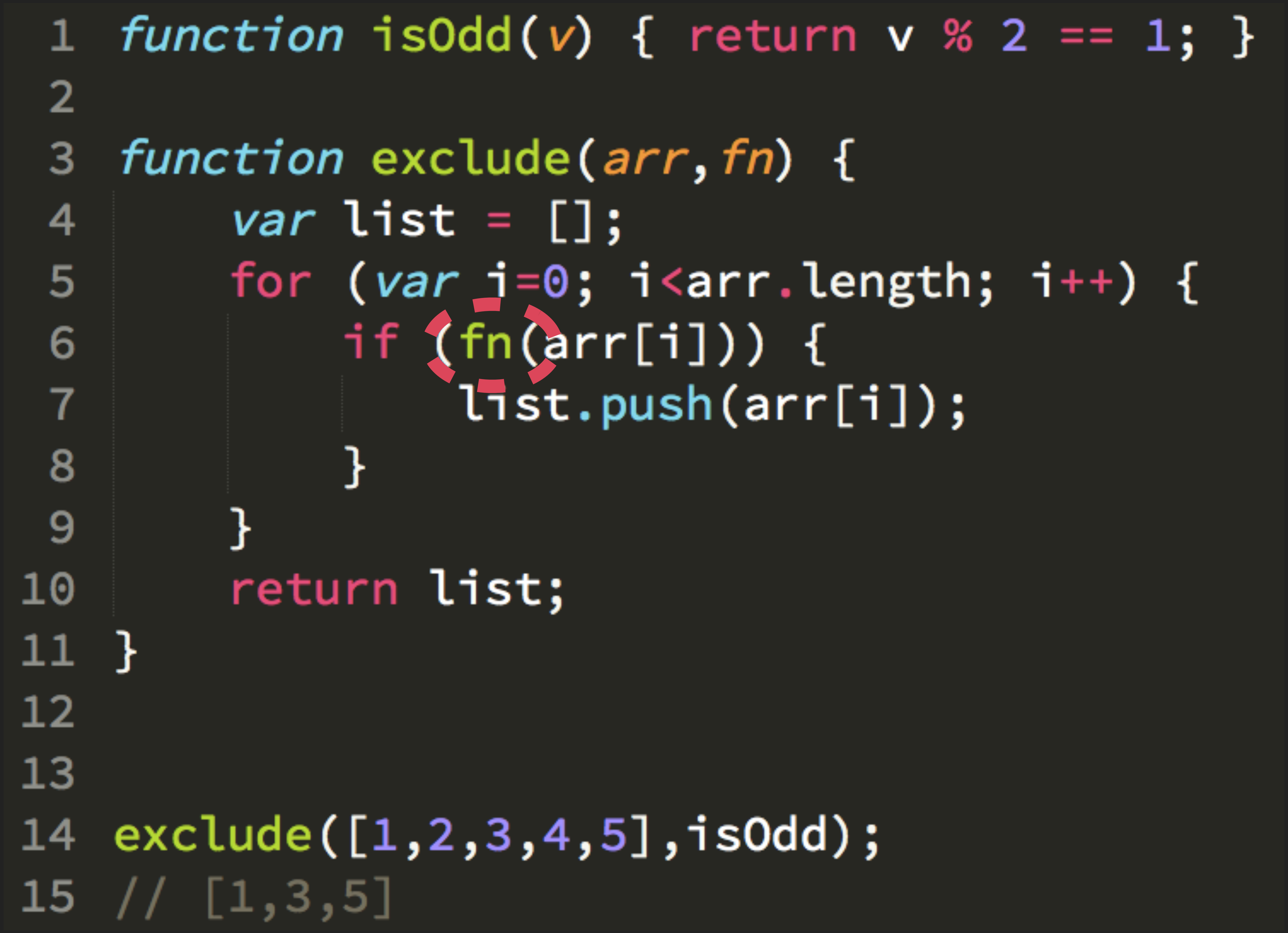
```javascript
function doubleIt(v) { return v * 2; }

function transform(arr,fn) {
    var list = [];
    for (var i=0; i<arr.length; i++) {
        list[i] = fn(arr[i]);
    }
    return list;
}


transform([1,2,3,4,5],doubleIt);
// [2,4,6,8,10]
```



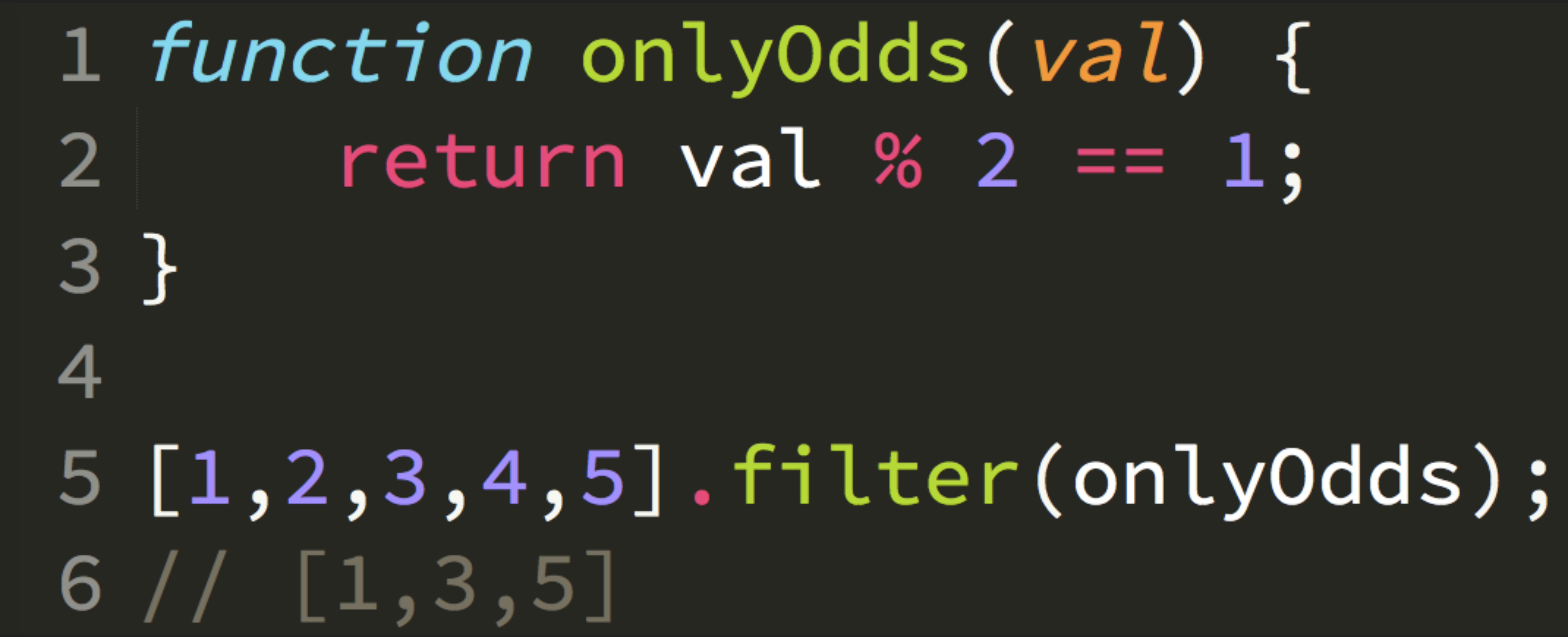
```javascript
function doubleIt(val) {
    return val * 2;
}

[1,2,3,4,5].map(doubleIt);
// [2,4,6,8,10]
```



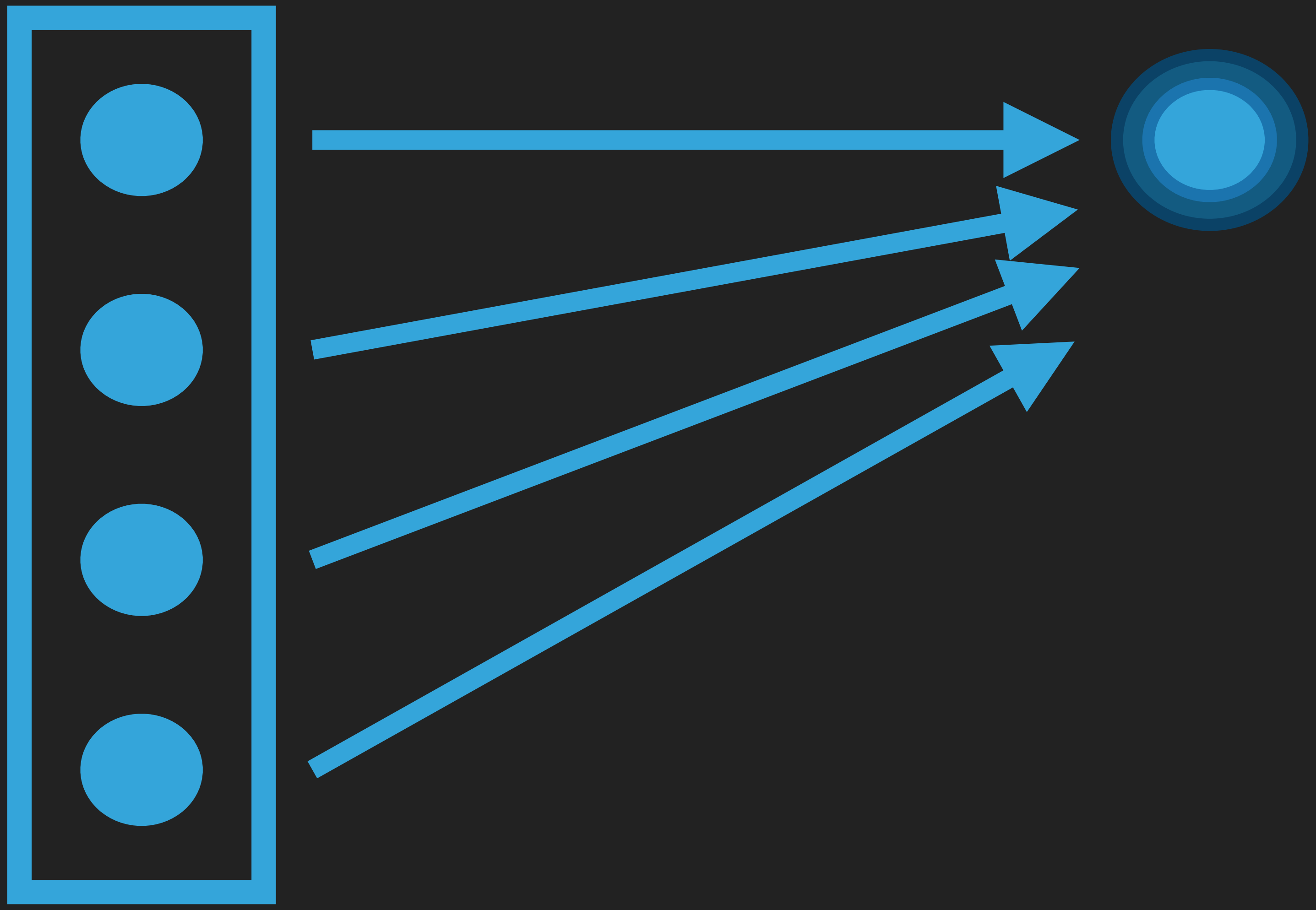
FILTER: EXCLUSION



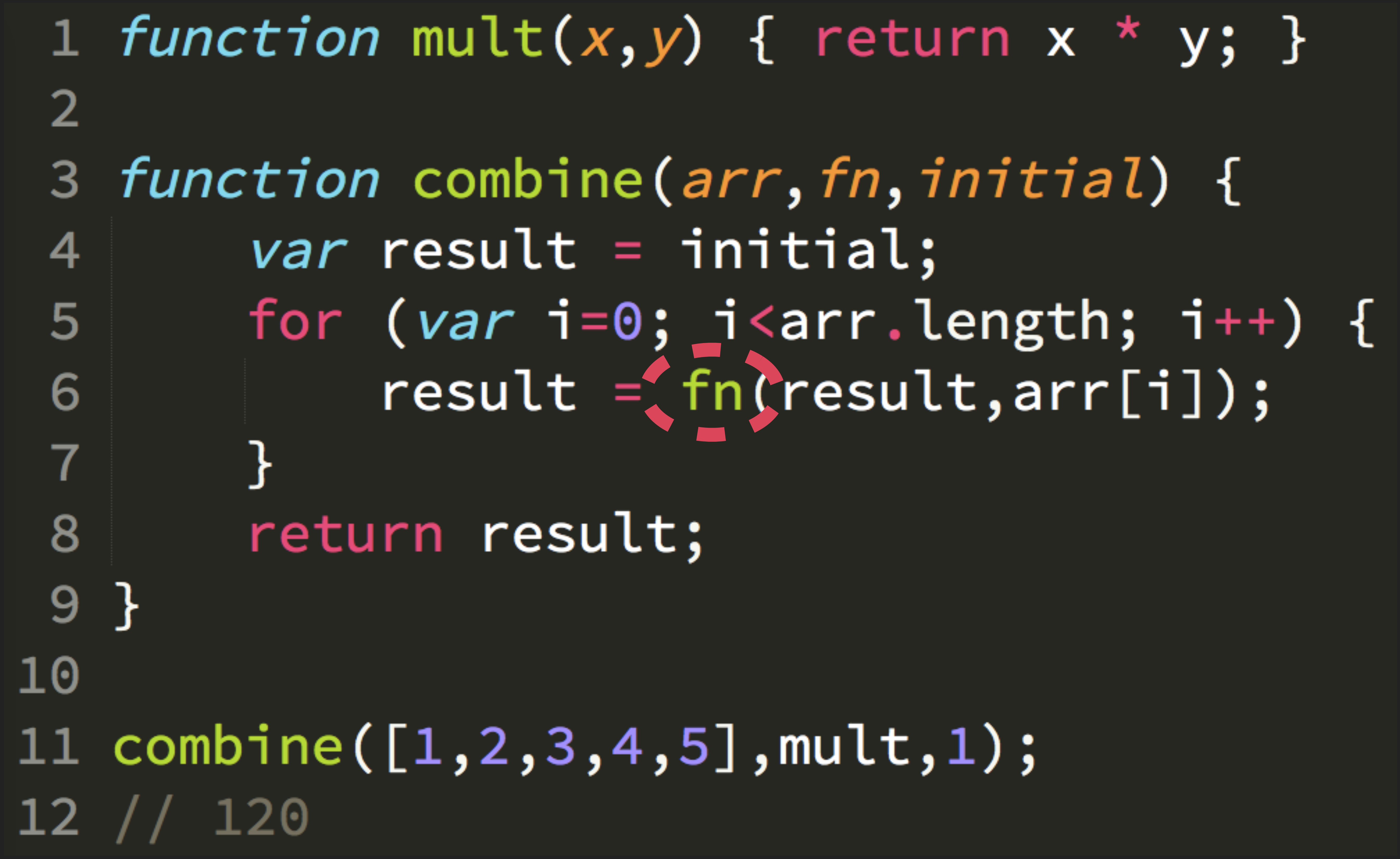
```
function isOdd(v) { return v % 2 == 1; }

function exclude(arr,fn) {
      var list = [];
      for (var i=0; i<arr.length; i++) {
            if (fn(arr[i])) {
                  list.push(arr[i]);
            }
      }
      return list;
}


exclude([1,2,3,4,5],isOdd);
// [1,3,5]
```



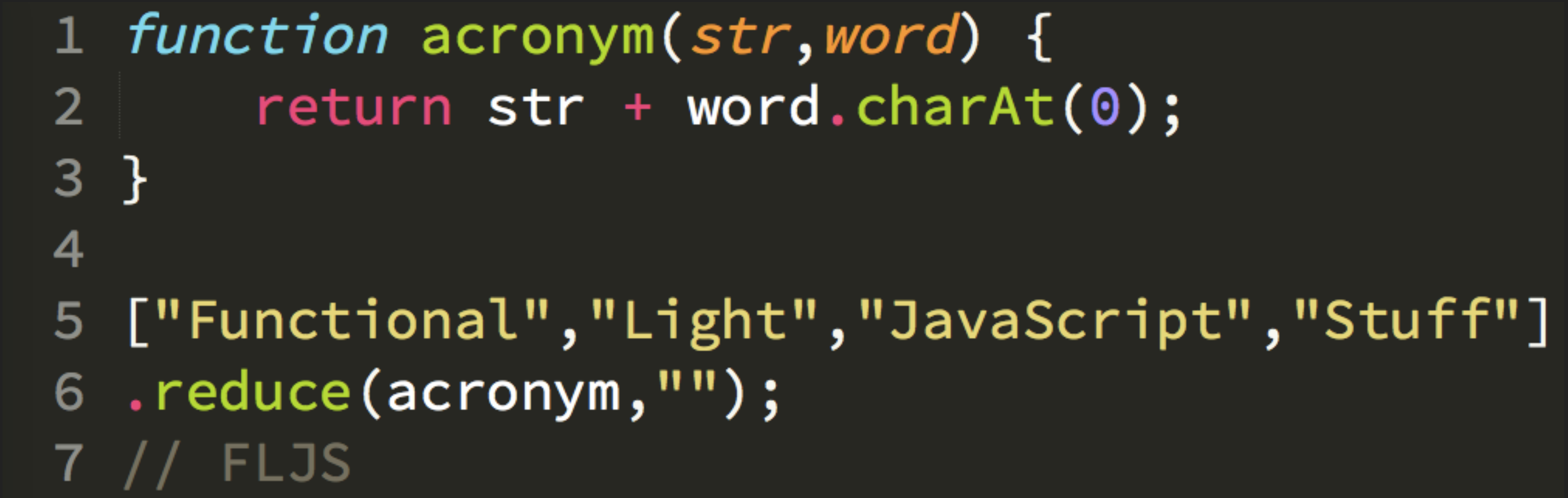
```javascript
function onlyOdds(val) {
    return val % 2 == 1;
}

[1,2,3,4,5].filter(onlyOdds);
// [1,3,5]
```



REDUCE: COMBINING



```
function mult(x,y) { return x * y; }

function combine(arr,fn,initial) {
    var result = initial;
    for (var i=0; i<arr.length; i++) {
        result = fn(result,arr[i]);
    }
    return result;
}

combine([1,2,3,4,5],mult,1);
// 120
```

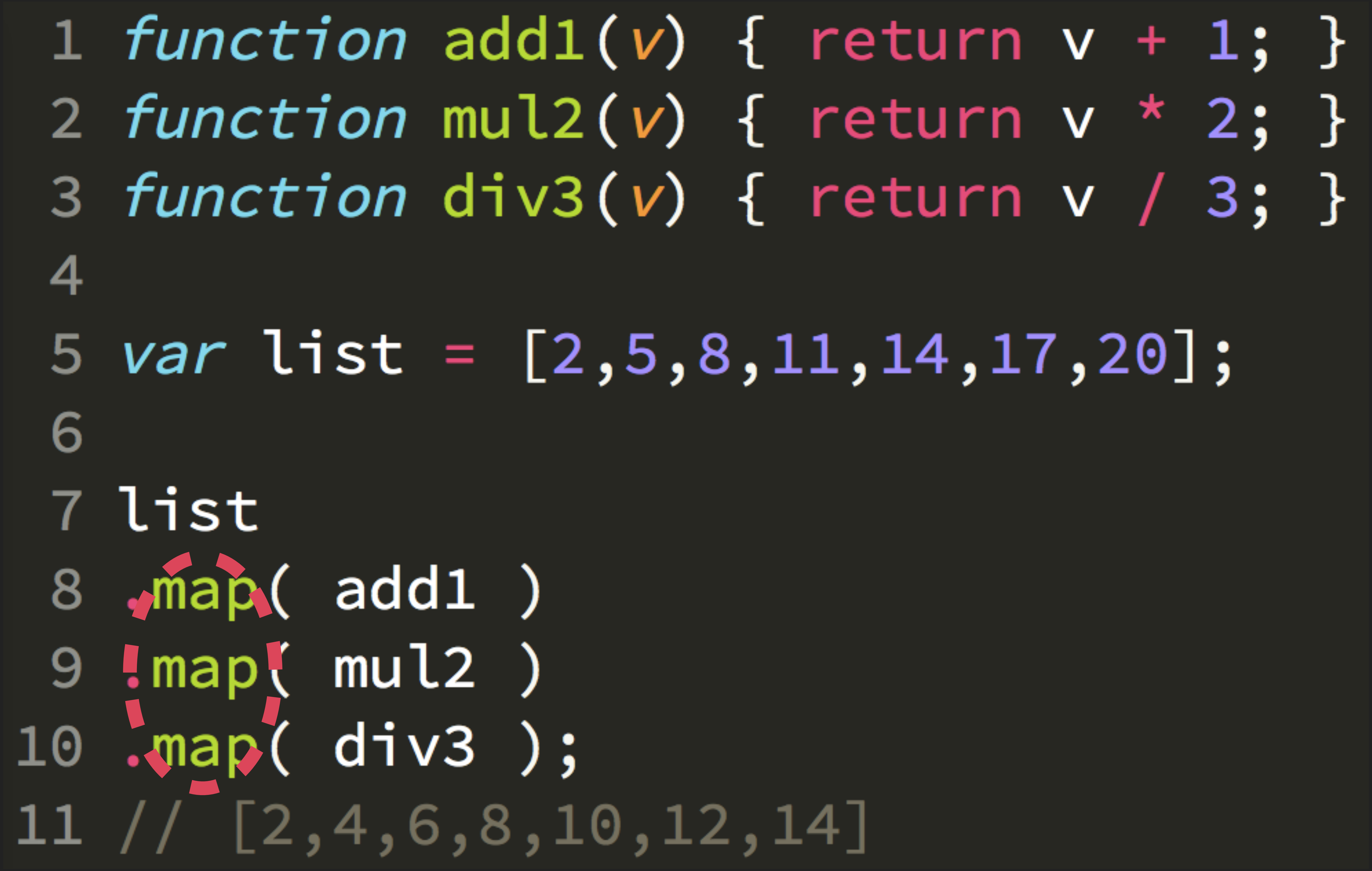


```javascript
function acronym(str,word) {
    return str + word.charAt(0);
}

["Functional","Light","JavaScript","Stuff"]
.reduce(acronym,"");
// FLJS
```

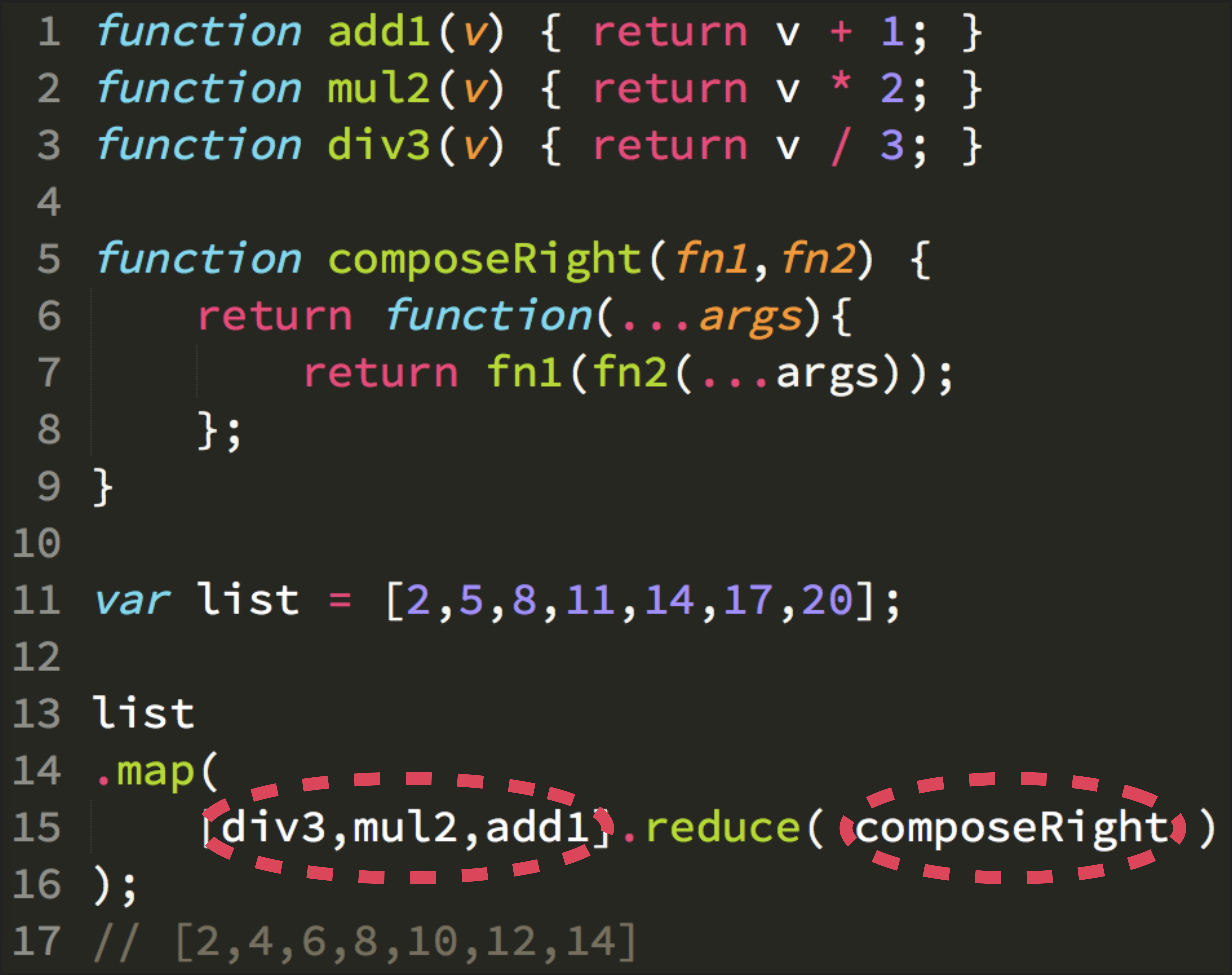
EXERCISE 4

EXTRA CREDIT

FUSION

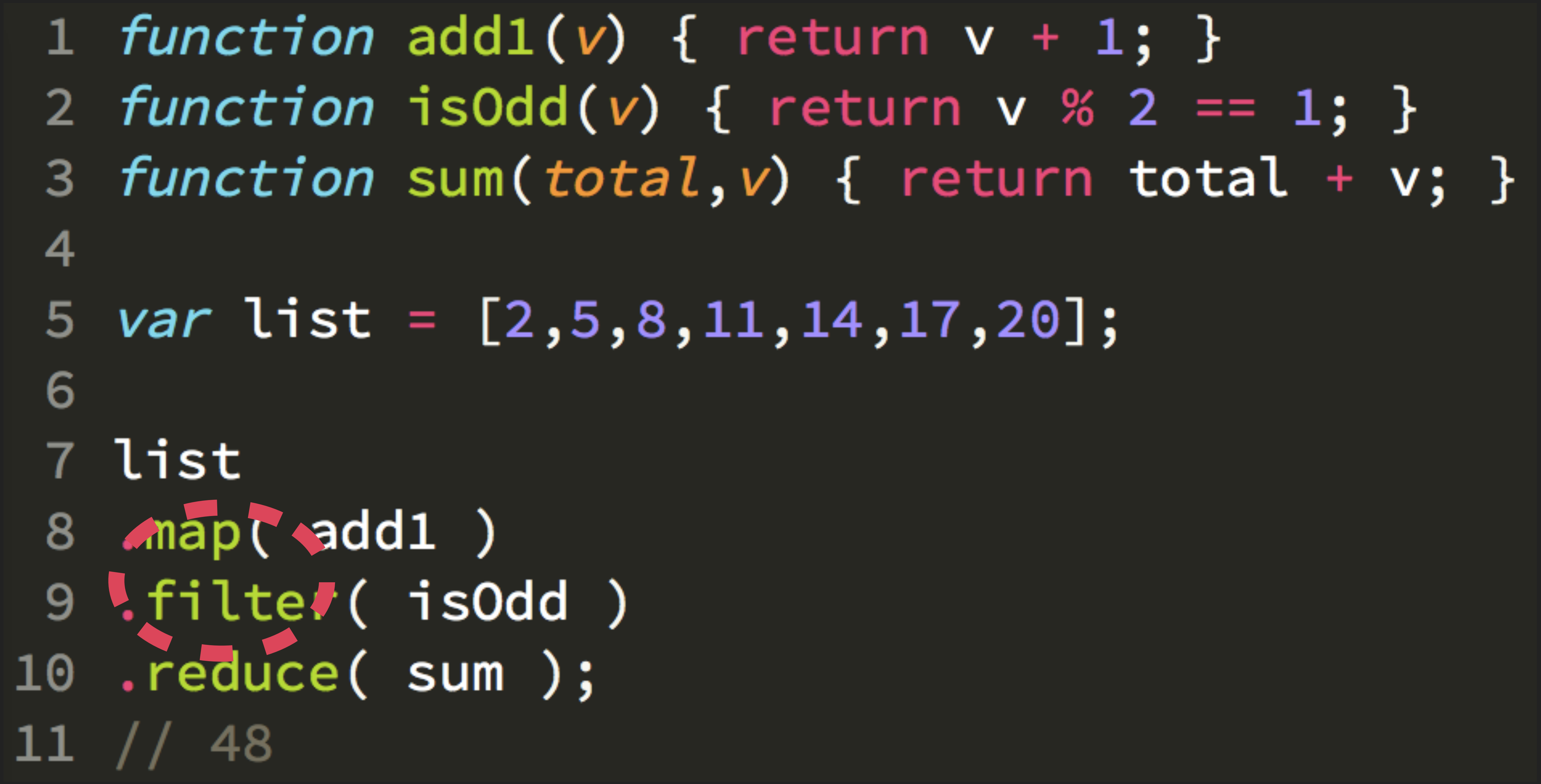


```
function add1(v) { return v + 1; }
function mul2(v) { return v * 2; }
function div3(v) { return v / 3; }

var list = [2,5,8,11,14,17,20];

list
.map( add1 )
.map( mul2 )
.map( div3 );
// [2,4,6,8,10,12,14]
```

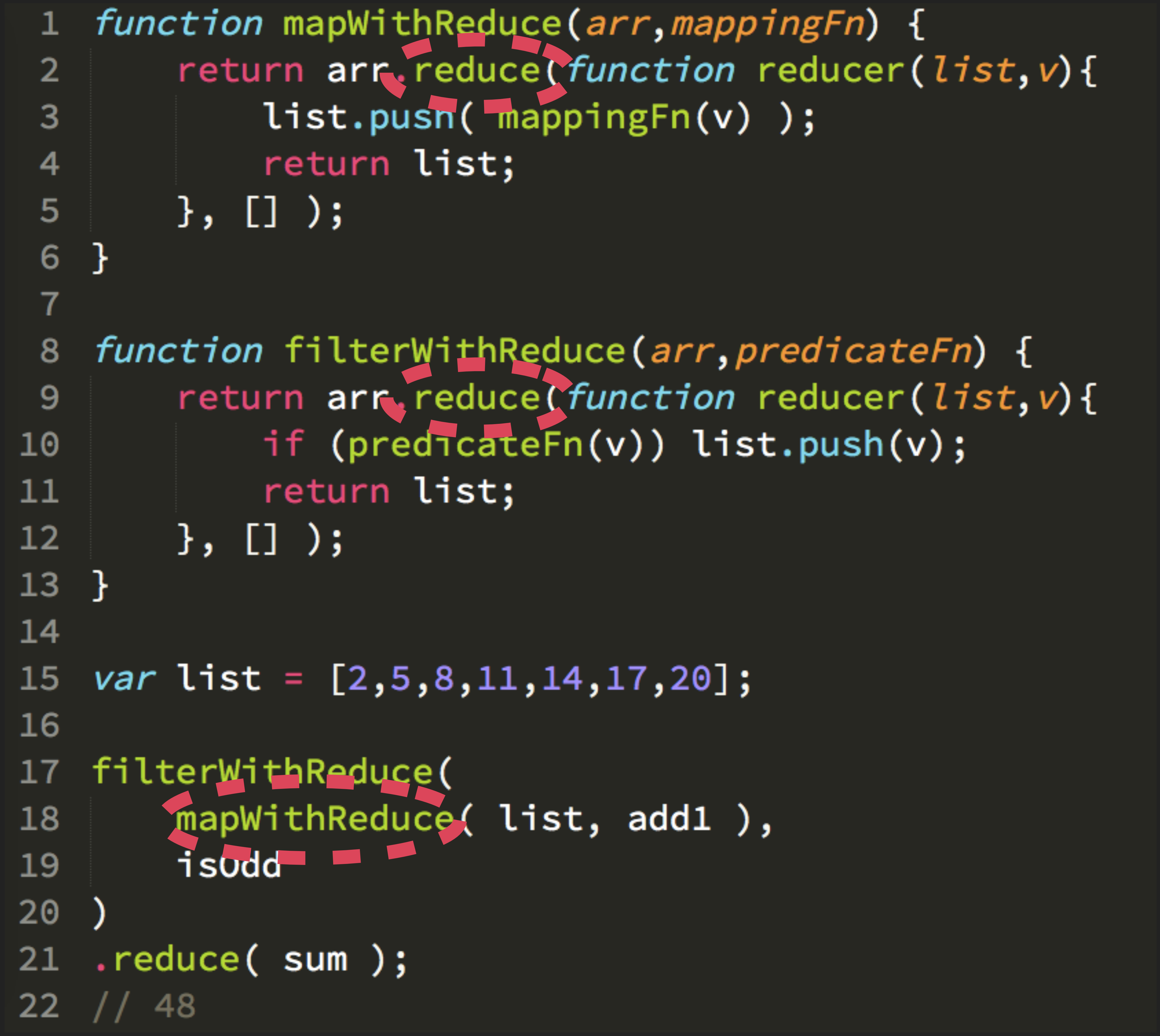


```
function add1(v) { return v + 1; }
function mul2(v) { return v * 2; }
function div3(v) { return v / 3; }

function composeRight(fn1,fn2) {
    return function(...args){
        return fn1(fn2(...args));
    };
}

var list = [2,5,8,11,14,17,20];

list
.map(
    [div3,mul2,add1].reduce( composeRight )
);
// [2,4,6,8,10,12,14]
```

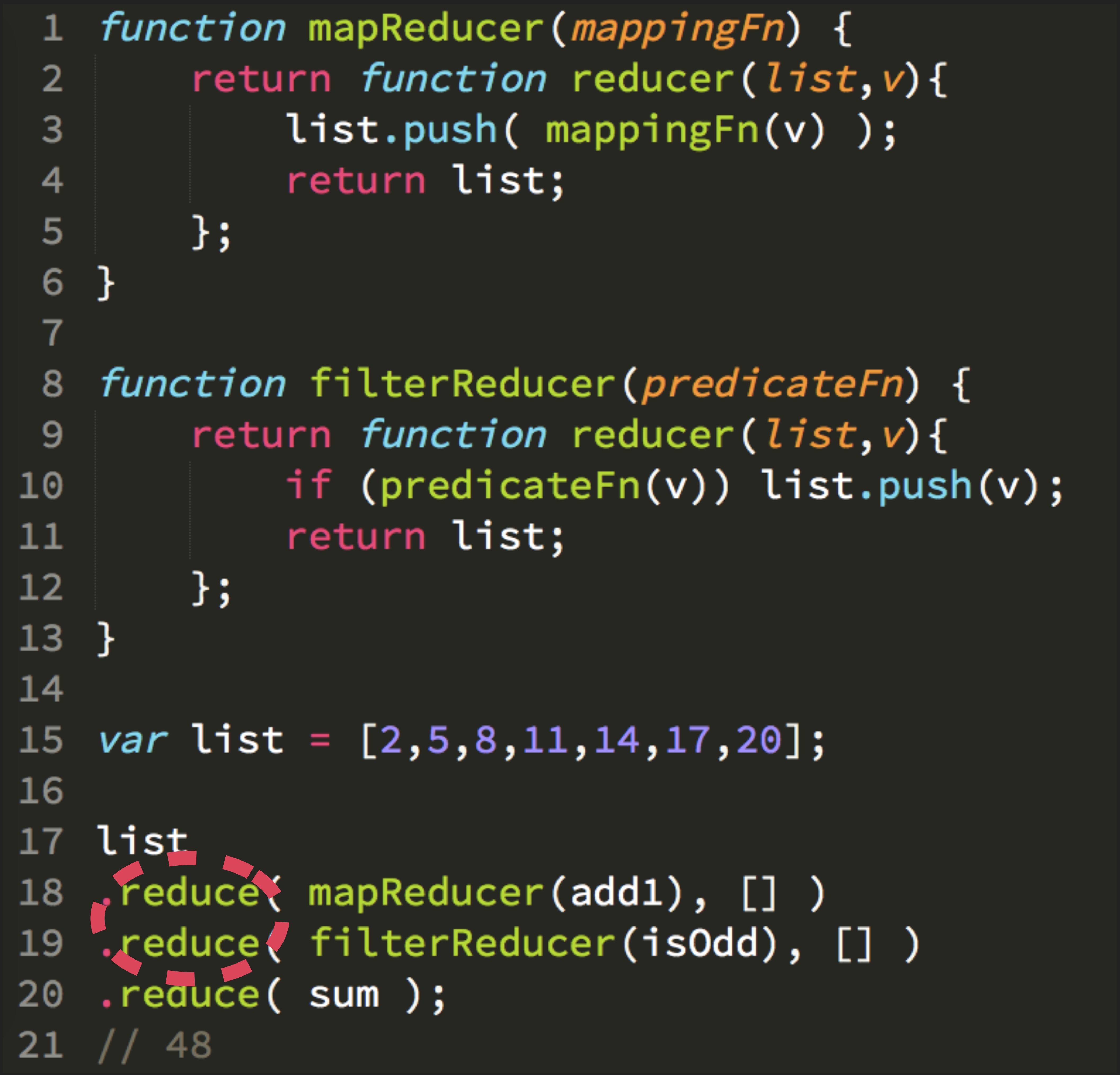
TRANSDUCE



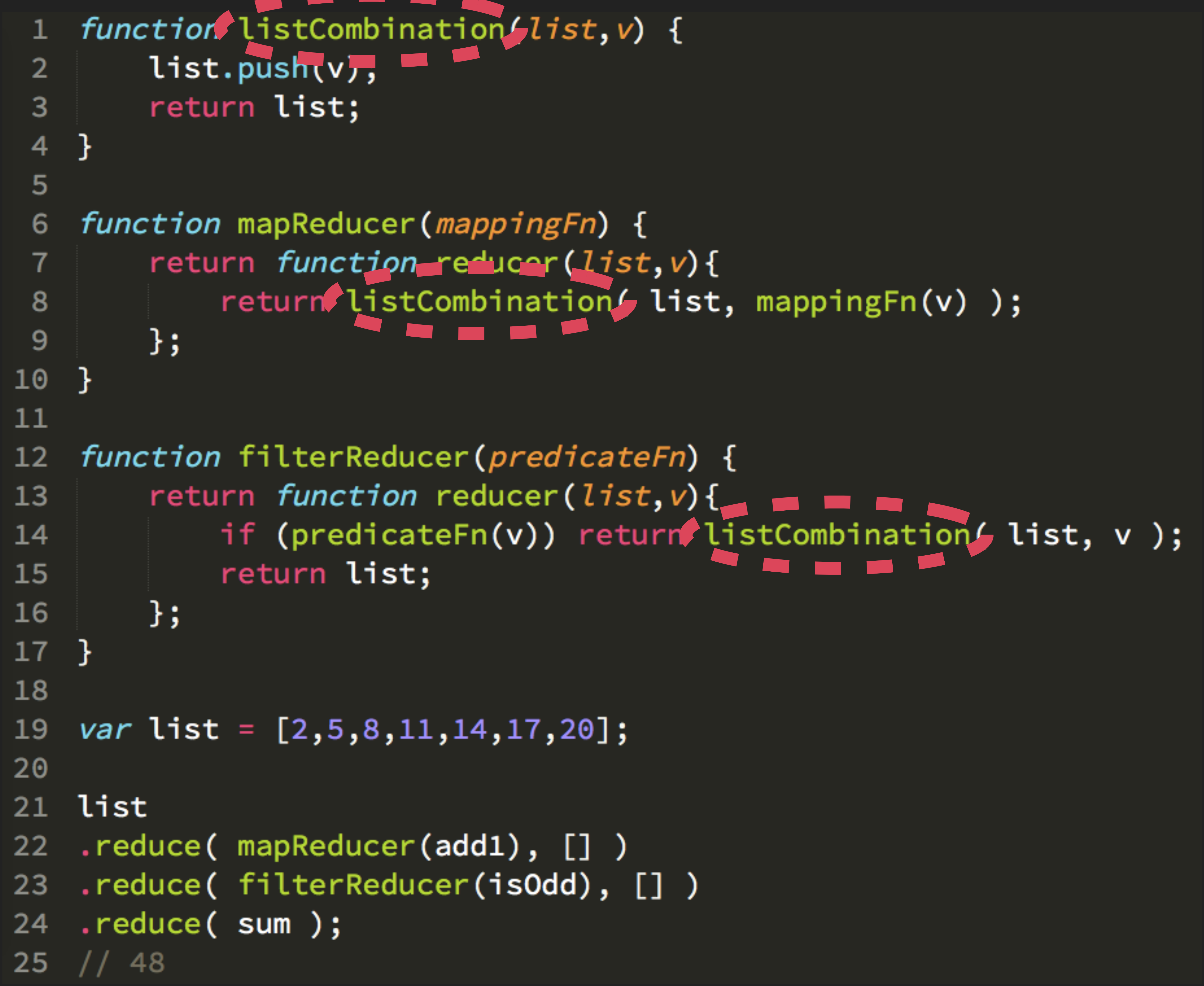
```javascript
function add1(v) { return v + 1; }
function isOdd(v) { return v % 2 == 1; }
function sum(total,v) { return total + v; }

var list = [2,5,8,11,14,17,20];

list
.map( add1 )
.filter( isOdd )
.reduce( sum );
// 48
```



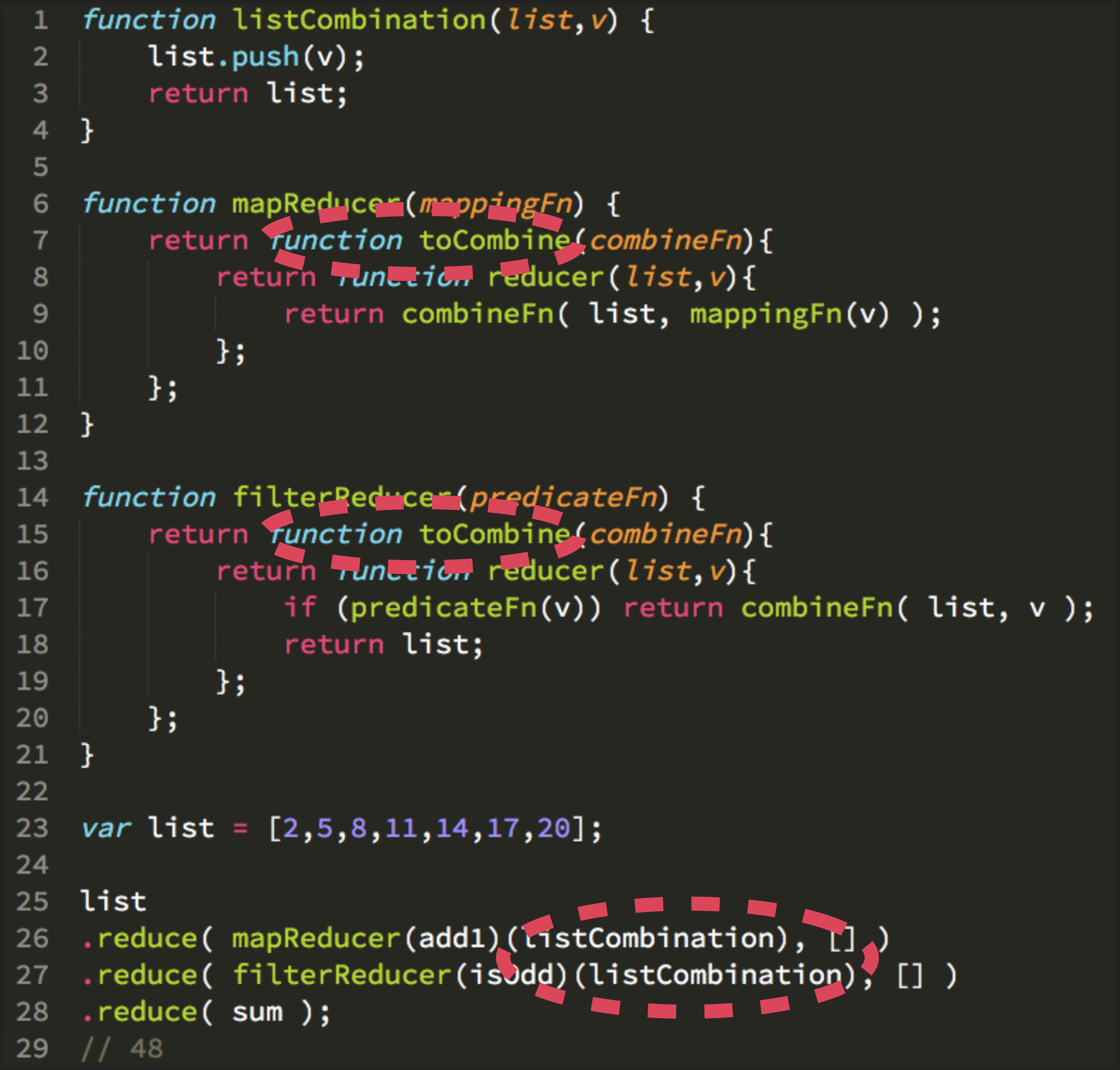
```
function mapWithReduce(arr,mappingFn) {
    return arr.reduce(function reducer(list,v){
        list.push( mappingFn(v) );
        return list;
    }, [] );
}

function filterWithReduce(arr,predicateFn) {
    return arr.reduce(function reducer(list,v){
        if (predicateFn(v)) list.push(v);
        return list;
    }, [] );
}

var list = [2,5,8,11,14,17,20];

filterWithReduce(
    mapWithReduce( list, add1 ),
    isOdd
)
.reduce( sum );
// 48
```



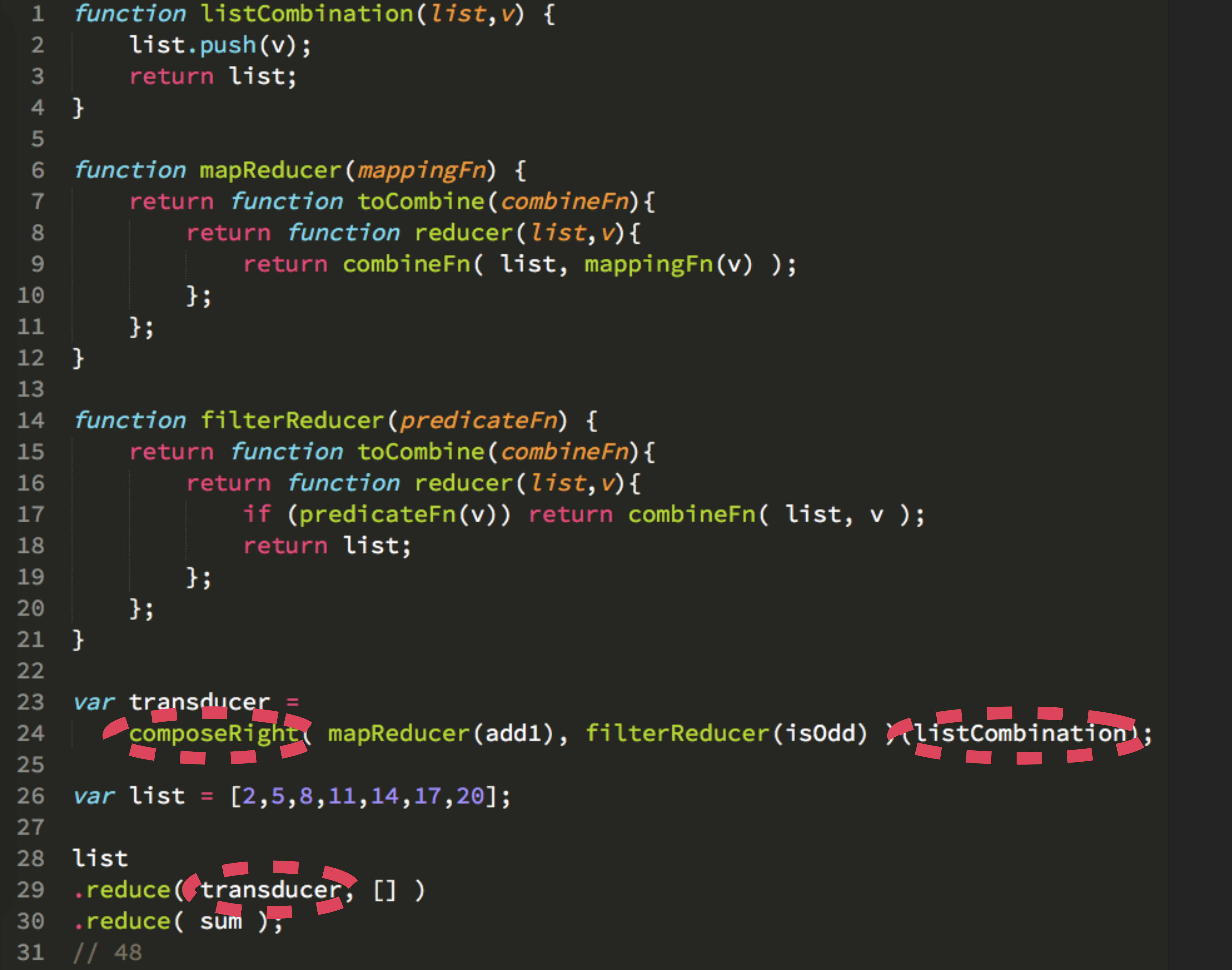
```
function mapReducer(mappingFn) {
    return function reducer(list,v){
        list.push( mappingFn(v) );
        return list;
    };
}

function filterReducer(predicateFn) {
    return function reducer(list,v){
        if (predicateFn(v)) list.push(v);
        return list;
    };
}

var list = [2,5,8,11,14,17,20];

list
.reduce( mapReducer(add1), [] )
.reduce( filterReducer(isOdd), [] )
.reduce( sum );
// 48
```



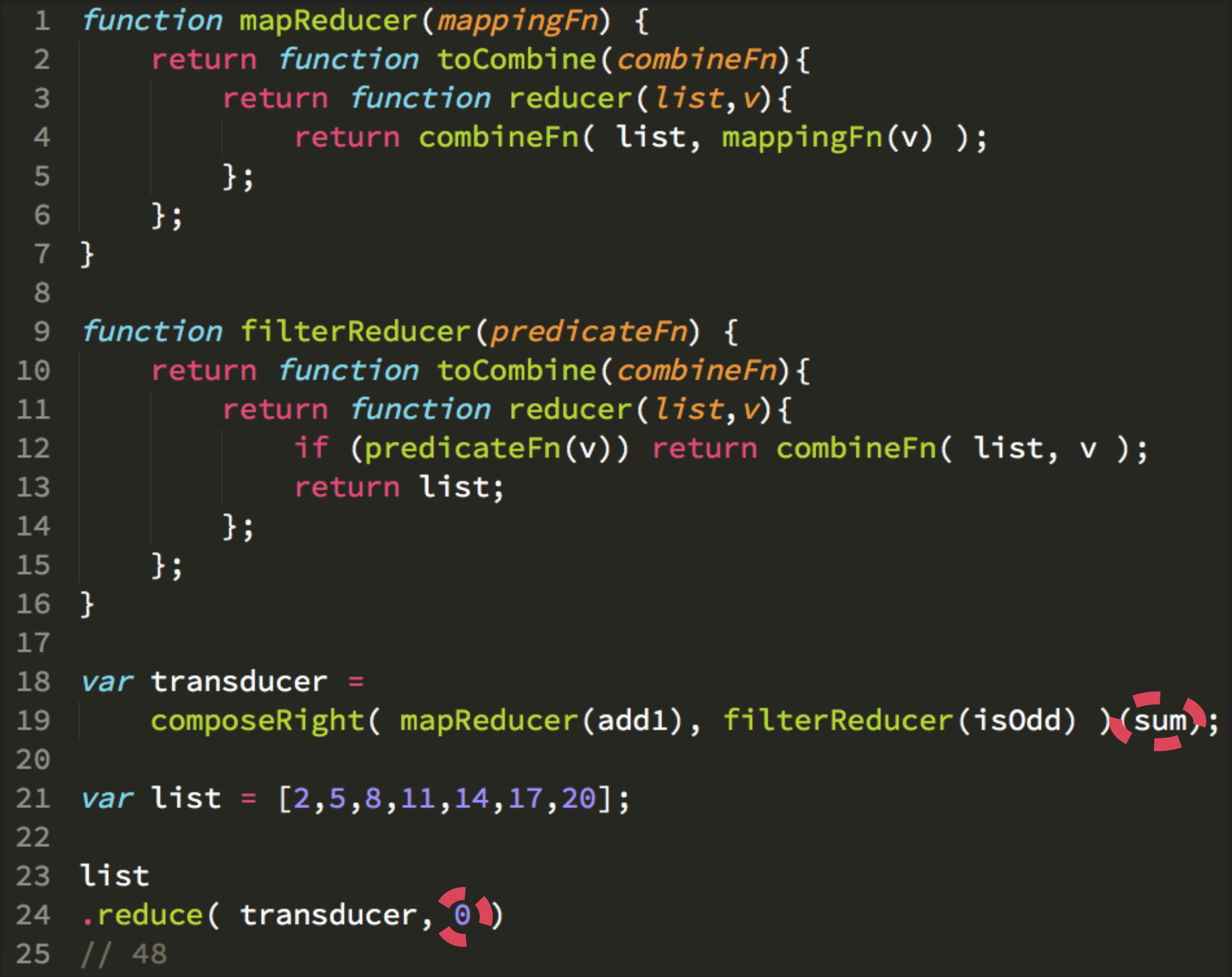
```
function listCombination(list,v) {
    list.push(v);
    return list;
}

function mapReducer(mappingFn) {
    return function reducer(list,v){
        return listCombination( list, mappingFn(v) );
    };
}

function filterReducer(predicateFn) {
    return function reducer(list,v){
        if (predicateFn(v)) return listCombination( list, v );
        return list;
    };
}

var list = [2,5,8,11,14,17,20];

list
.reduce( mapReducer(add1), [] )
.reduce( filterReducer(isOdd), [] )
.reduce( sum );
// 48
```



```
function listCombination(list,v) {
    list.push(v);
    return list;
}

function mapReducer(mappingFn) {
    return function toCombine(combineFn){
        return function reducer(list,v){
            return combineFn( list, mappingFn(v) );
        };
    };
}

function filterReducer(predicateFn) {
    return function toCombine(combineFn){
        return function reducer(list,v){
            if (predicateFn(v)) return combineFn( list, v );
            return list;
        };
    };
}

var list = [2,5,8,11,14,17,20];

list
.reduce( mapReducer(add1)(listCombination), [] )
.reduce( filterReducer(isOdd)(listCombination), [] )
.reduce( sum );
// 48
```



```javascript
function listCombination(list,v) {
    list.push(v);
    return list;
}

function mapReducer(mappingFn) {
    return function toCombine(combineFn){
        return function reducer(list,v){
            return combineFn( list, mappingFn(v) );
        };
    };
}

function filterReducer(predicateFn) {
    return function toCombine(combineFn){
        return function reducer(list,v){
            if (predicateFn(v)) return combineFn( list, v );
            return list;
        };
    };
}

var transducer =
    composeRight( mapReducer(add1), filterReducer(isOdd) )(listCombination);

var list = [2,5,8,11,14,17,20];

list
.reduce( transducer, [] )
.reduce( sum );
// 48
```



```javascript
function mapReducer(mappingFn) {
    return function toCombine(combineFn){
        return function reducer(list,v){
            return combineFn( list, mappingFn(v) );
        };
    };
}

function filterReducer(predicateFn) {
    return function toCombine(combineFn){
        return function reducer(list,v){
            if (predicateFn(v)) return combineFn( list, v );
            return list;
        };
    };
}

var transducer =
    composeRight( mapReducer(add1), filterReducer(isOdd) )(sum);

var list = [2,5,8,11,14,17,20];

list
.reduce( transducer, 0)
// 48
```

RECAP:

- Pure Functions (~~side effects~~)
- Composition
- Immutability
- Closure
- Lists (map, filter, reduce)
  (fusion, transducing)

THANKS!!!!

KYLE SIMPSON GETIFY@GMAIL.COM

FUNCTIONAL-LIGHT JS