Developer Student Clubs

Getting Started with JavaScript

Developer Student Clubs

Table of Content

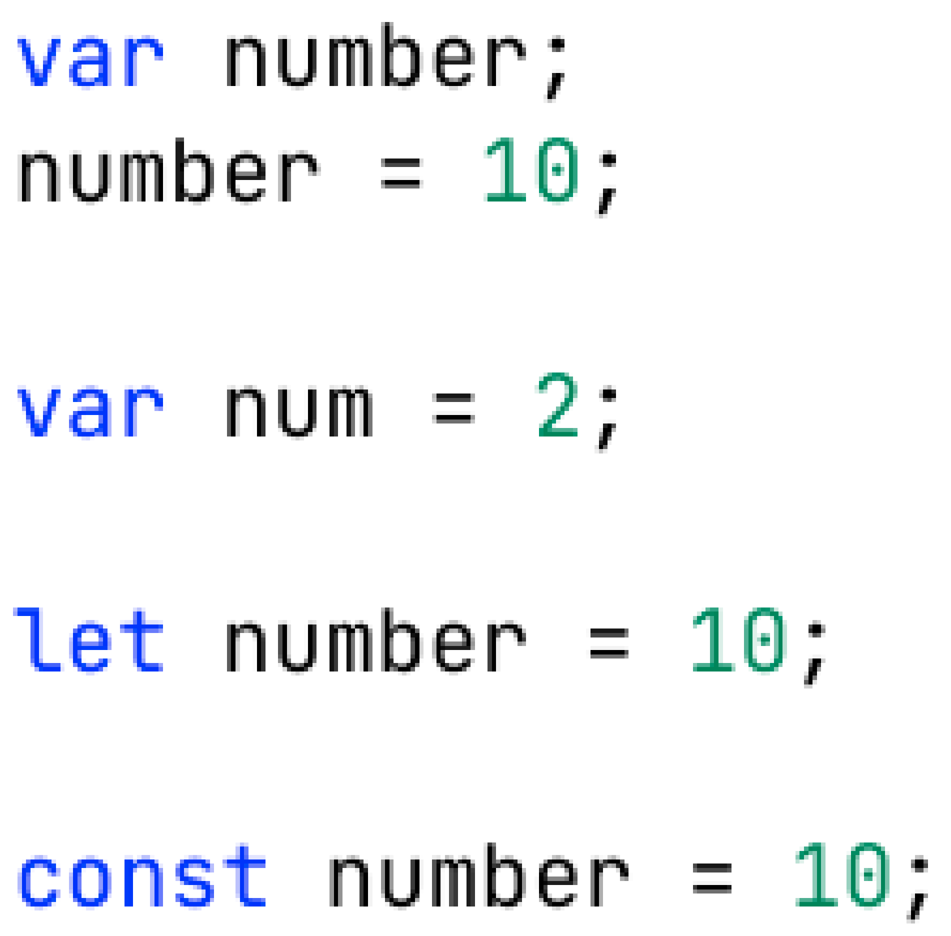
- Introduction
- Variable
- Conditional and Loops
- Function
- Objects and Arrays

What is JavaScript

JavaScript is the programming language that is used to create dynamic and interactive web content like applications and browsers

Variable

Variable - A JavaScript variable is simply a name of storage location.



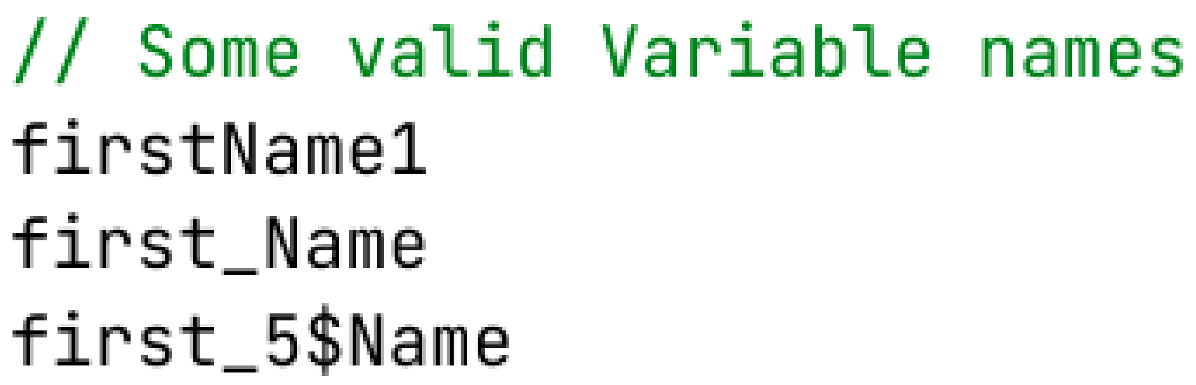
```
var number;
number = 10;

var num = 2;

let number = 10;

const number = 10;
```

How should I name my variables?

Naming Convention : camel-cased - Used for better understanding of variable.
Examples : userName , firstName

Valid identifiers in JS :
There are some rules while declaring a JavaScript variable (also known as identifiers).

- Name only contains letters, numbers, underscore(_) and dollar($)
- Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.



```
// Some valid Variable names
firstName1
first_Name
first_5$Name
```

Template Literals

Template literals are literals delimited with backtick (`) characters, allowing for multi-line strings, string interpolation with embedded expressions, and special constructs called tagged templates.

```
const firstName = 'Unga'
const lastName  = 'Bunga'

const intro = "My name is Unga Bunga"

const intro_1 = "My name is "+ firstName + " " + lastName

const intro_2 = `My name is ${firstName} ${lastName}`

console.log(intro)
console.log(intro_1)
console.log(introUsingTemplateLiterals)
```

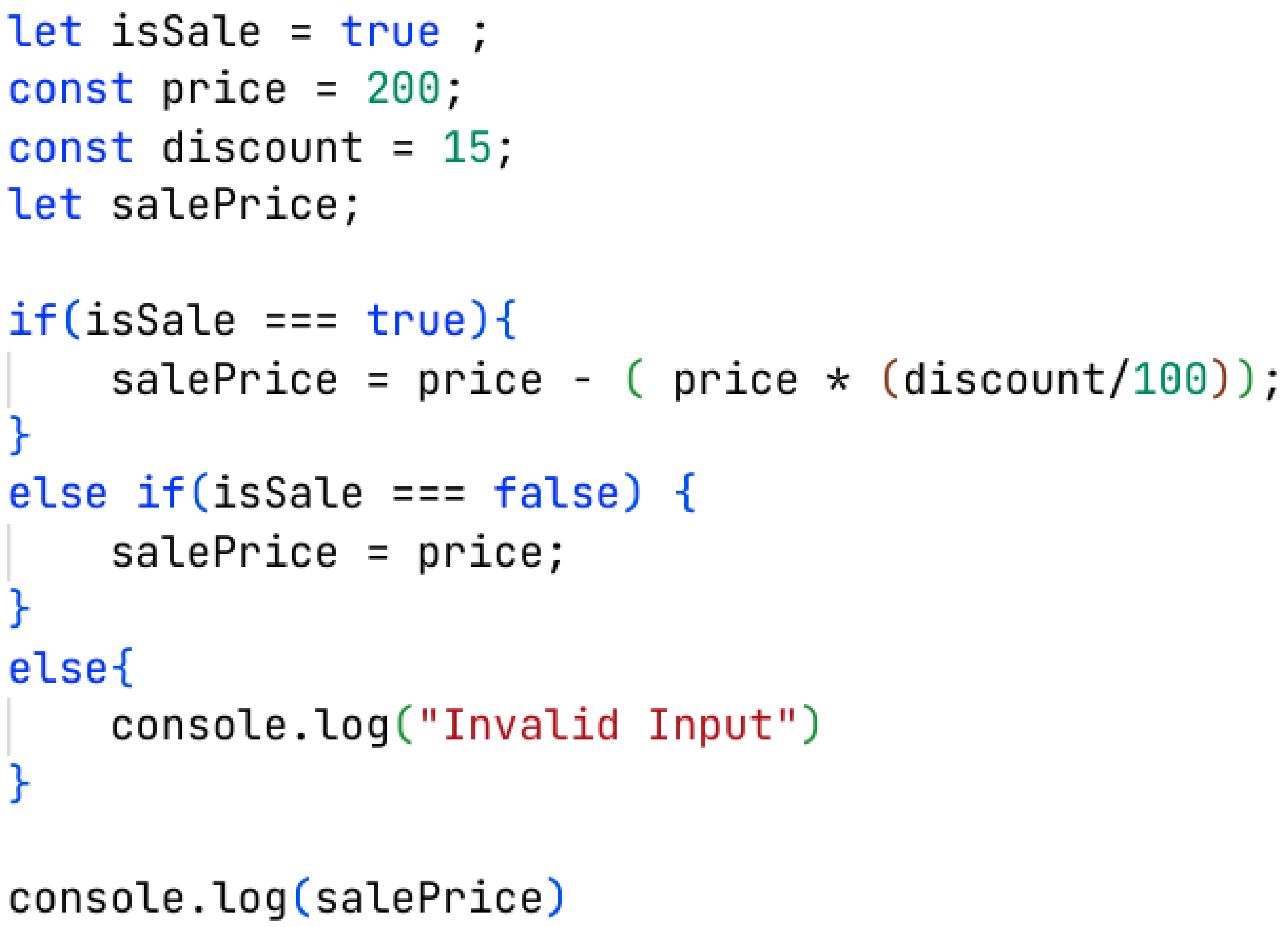
Conditionals

Conditional statements control behavior in JavaScript and determine whether or not pieces of code can run.

Three types of conditional statements :

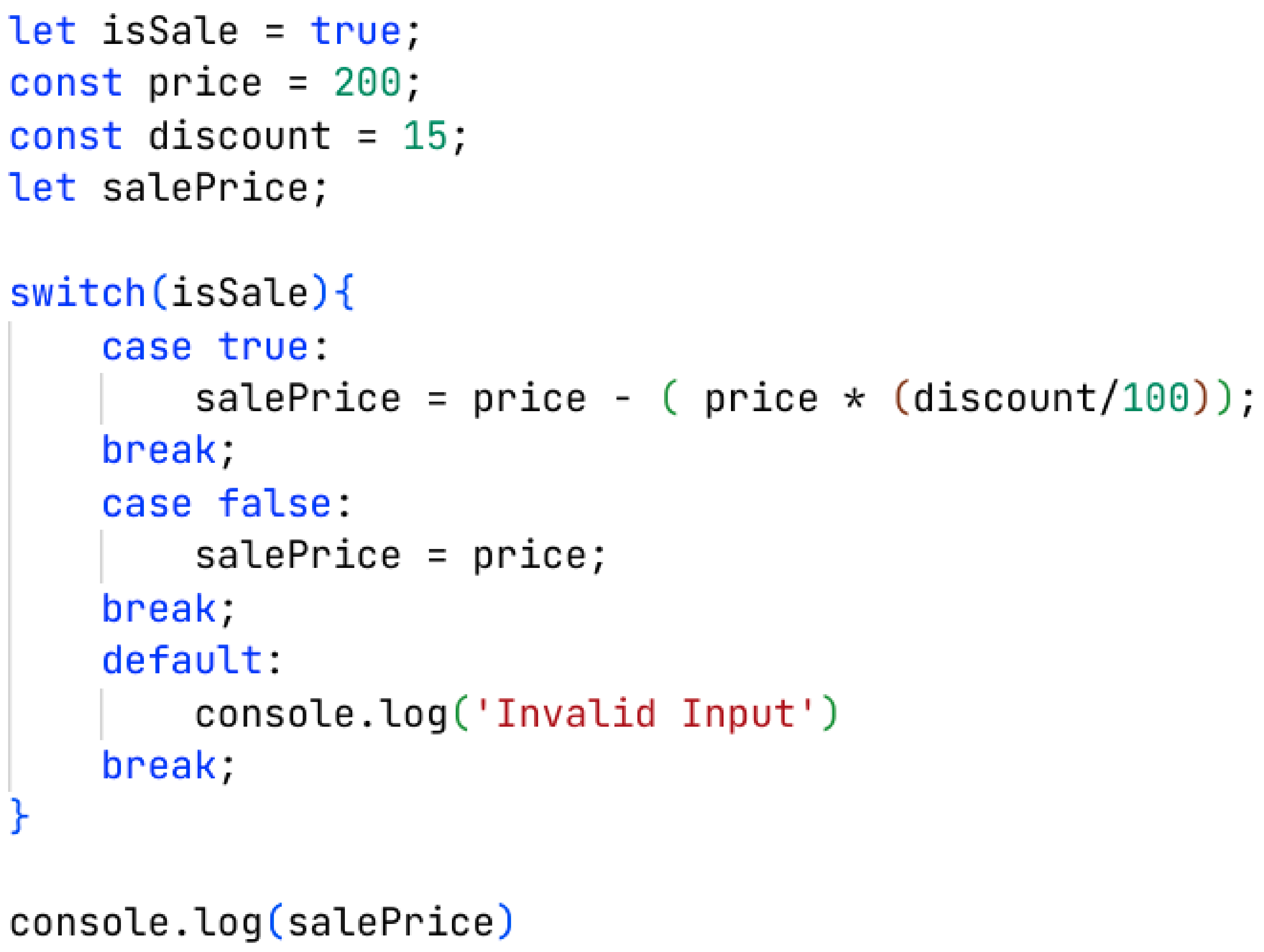
- If Else Statements
- Switch Case
- Ternary Operator

If Else Statements



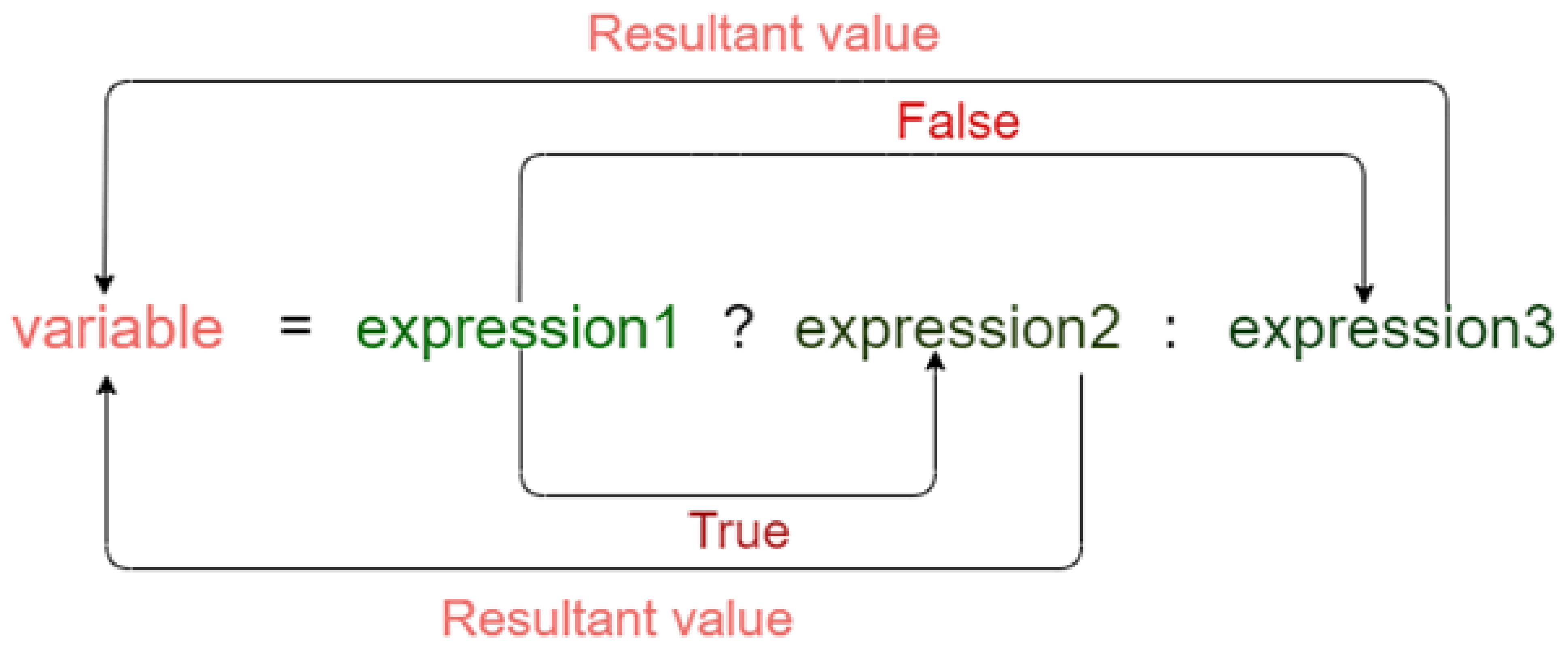
```javascript
let isSale = true ;
const price = 200;
const discount = 15;
let salePrice;

if(isSale === true){
    salePrice = price - ( price * (discount/100));
}
else if(isSale === false) {
    salePrice = price;
}
else{
    console.log("Invalid Input")
}

console.log(salePrice)
```

Switch Case



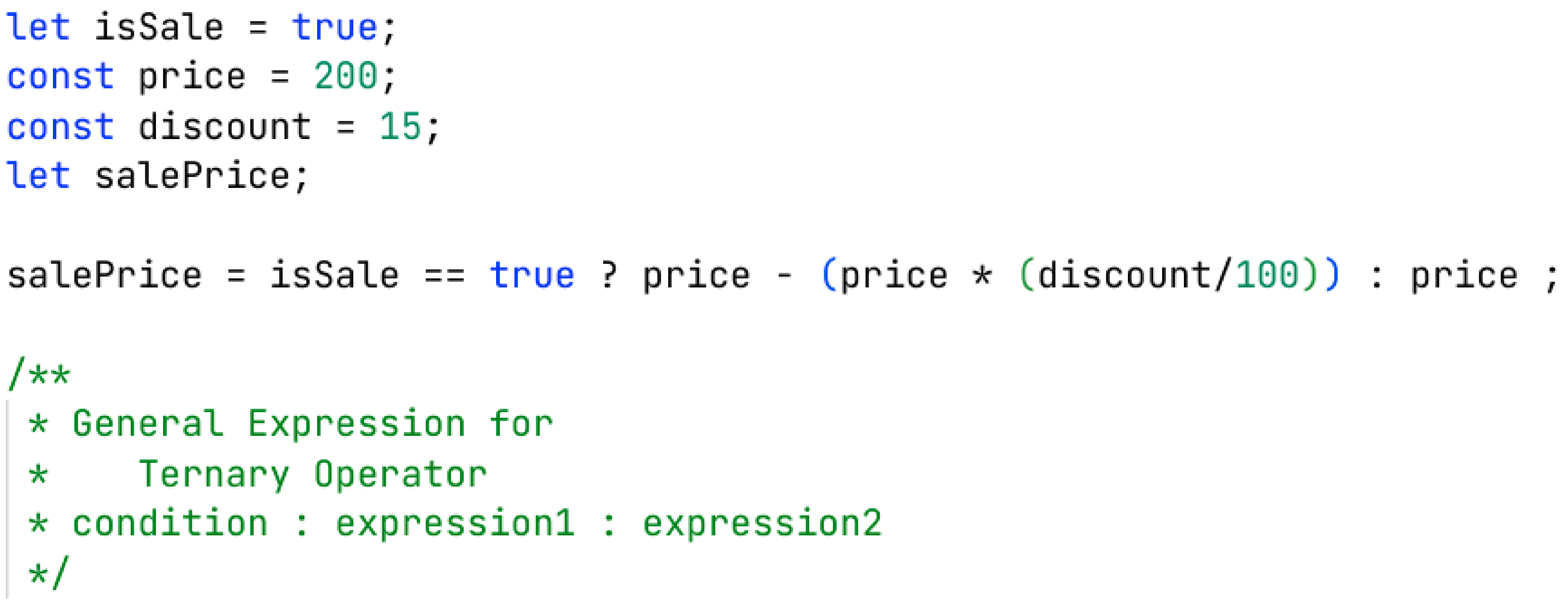
```
let isSale = true;
const price = 200;
const discount = 15;
let salePrice;

switch(isSale){
    case true:
        salePrice = price - ( price * (discount/100));
    break;
    case false:
        salePrice = price;
    break;
    default:
        console.log('Invalid Input')
    break;
}

console.log(salePrice)
```

Ternary Operator



Ternary Operator

Developer Student Clubs



```
let isSale = true;
const price = 200;
const discount = 15;
let salePrice;

salePrice = isSale == true ? price - (price * (discount/100)) : price ;

/**
 * General Expression for
 *    Ternary Operator
 * condition : expression1 : expression2
 */
```

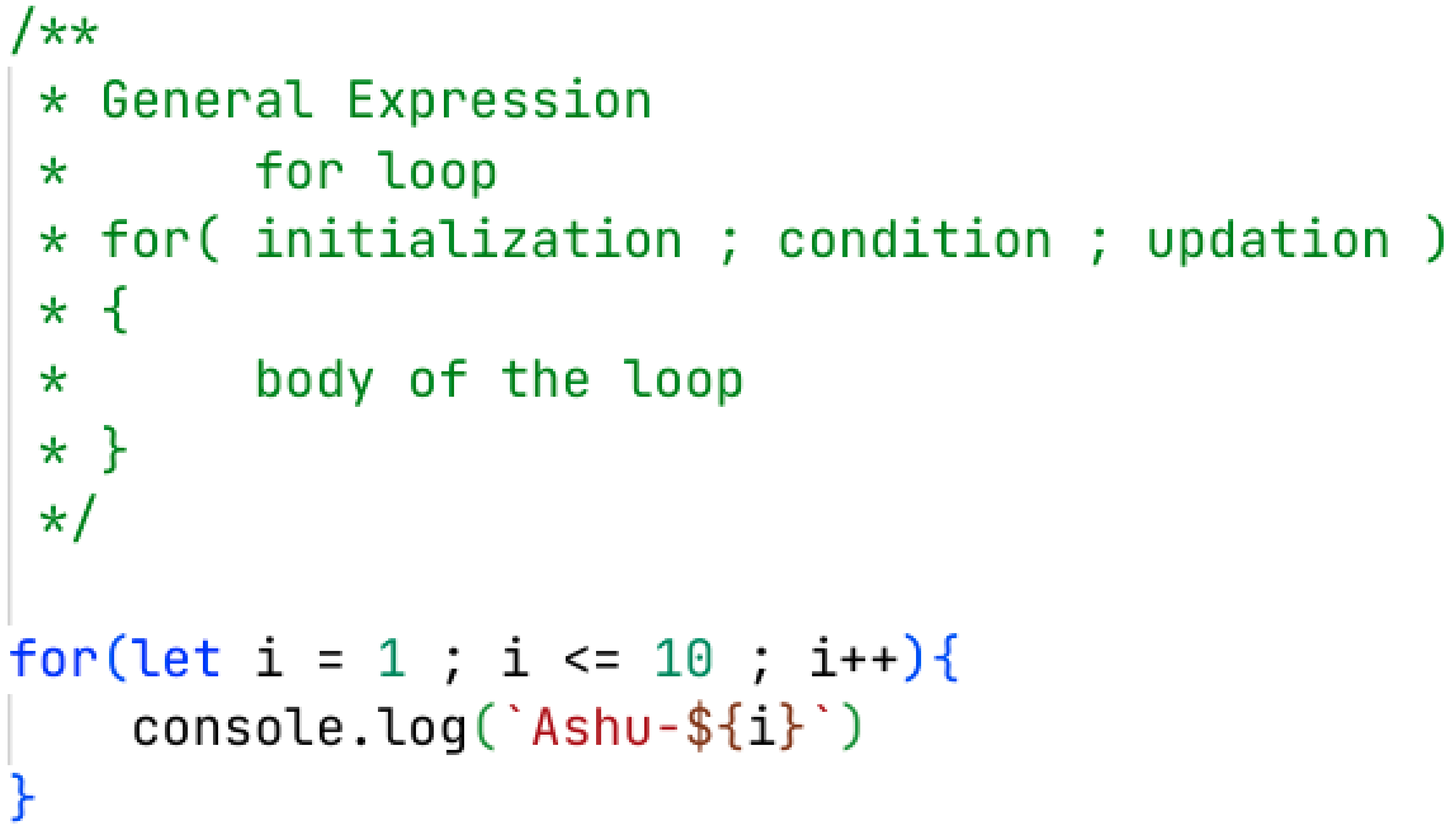
Loops

Looping in programming languages is a feature that facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true.

Three types of looping statements :

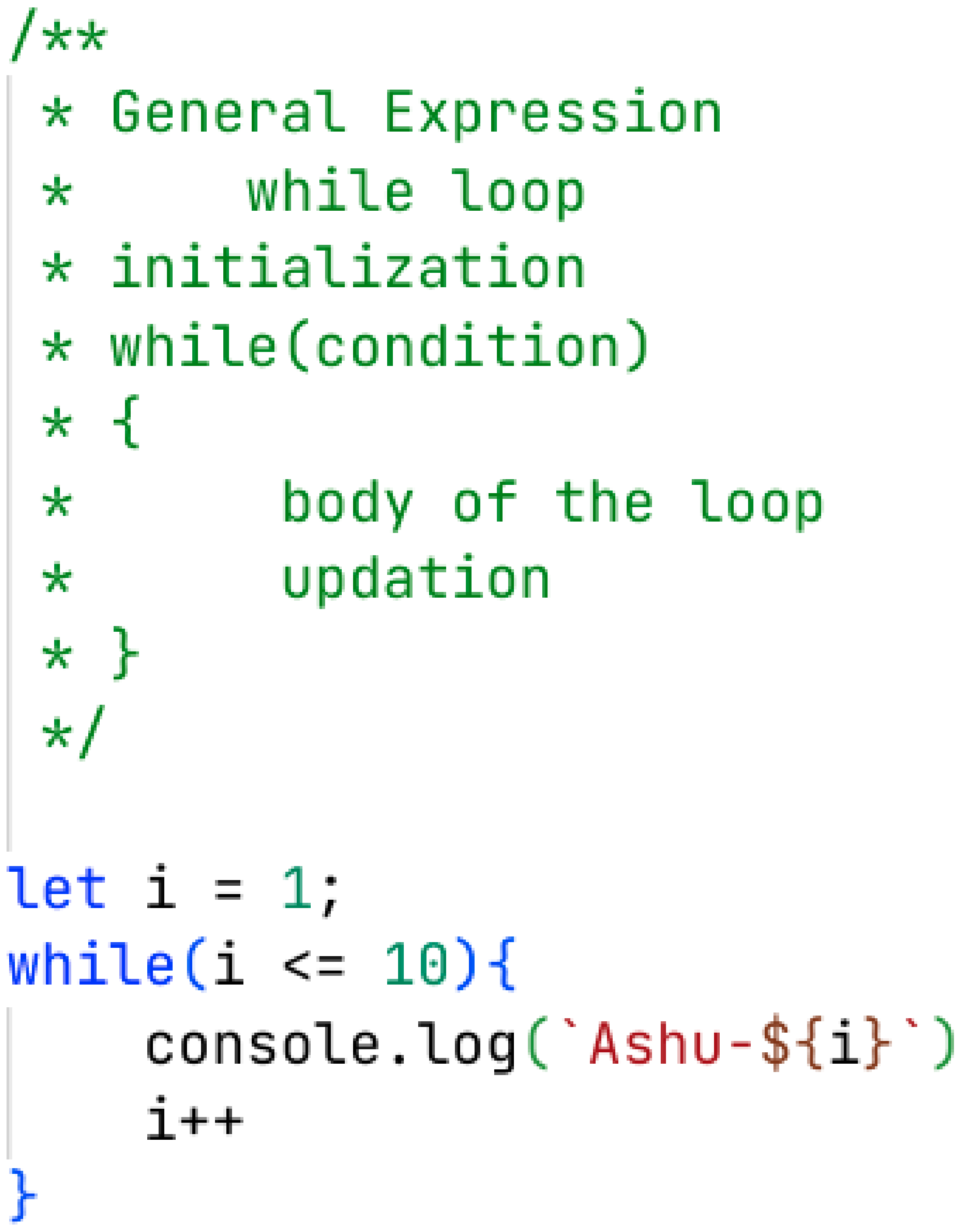
- for
- while
- do while

for Loop



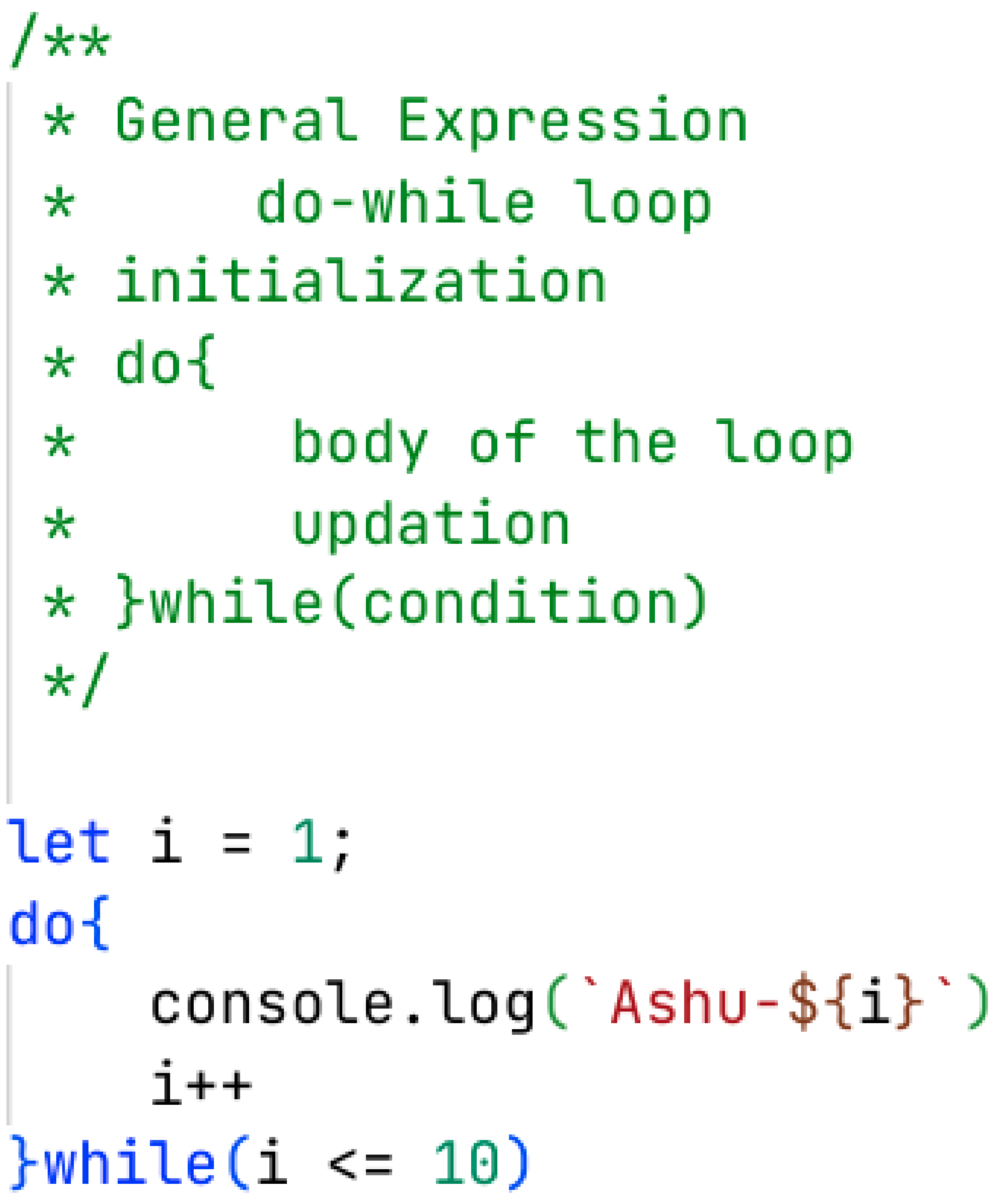
```
/**
 * General Expression
 *    for loop
 * for( initialization ; condition ; updation )
 * {
 *    body of the loop
 * }
 */

for(let i = 1 ; i <= 10 ; i++){
  console.log(`Ashu-${i}`)
}
```

while Loop



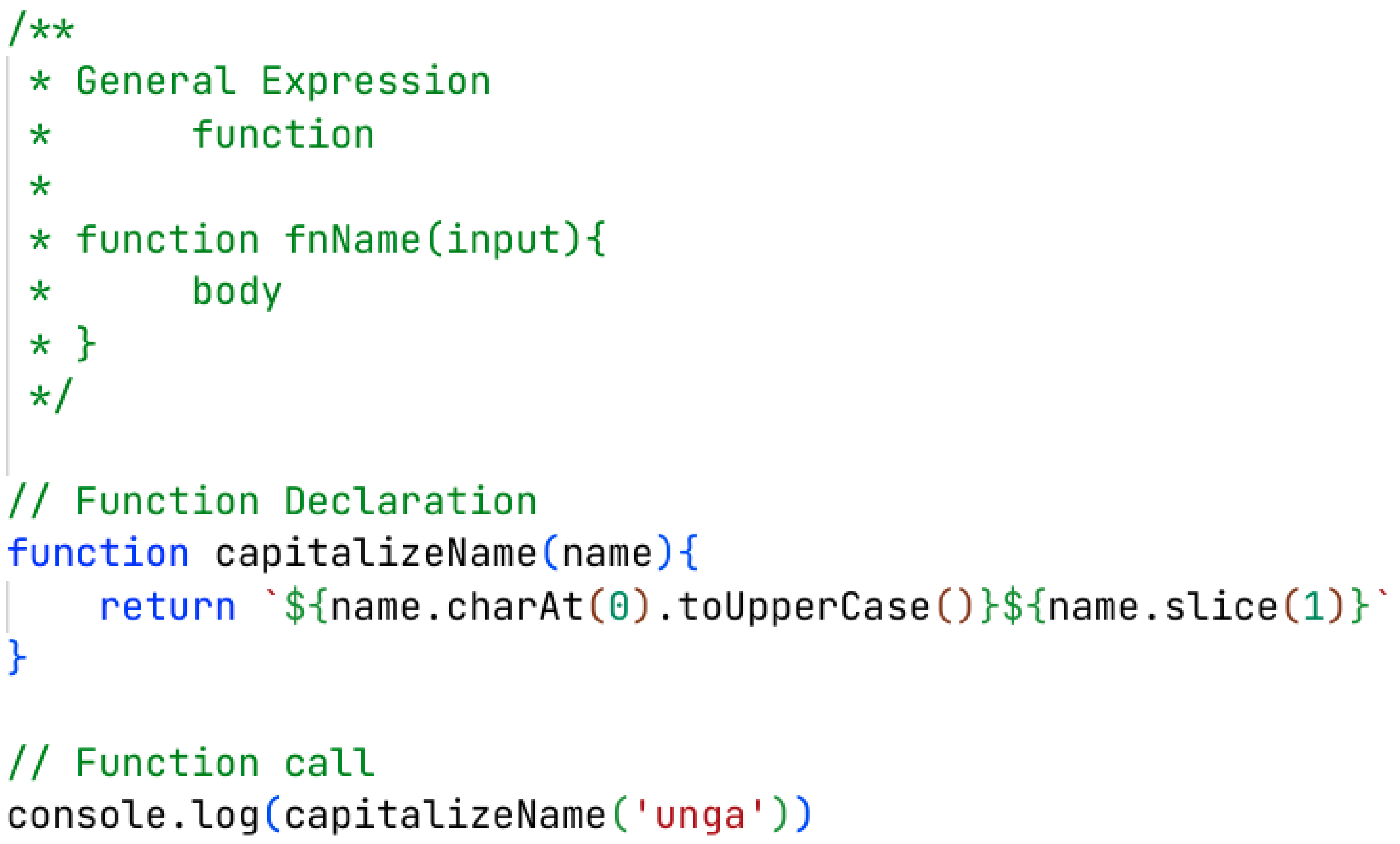
```
/**
 * General Expression
 *     while loop
 * initialization
 * while(condition)
 * {
 *      body of the loop
 *      updation
 * }
 */

let i = 1;
while(i <= 10){
    console.log(`Ashu-${i}`)
    i++
}
```

do while Loop



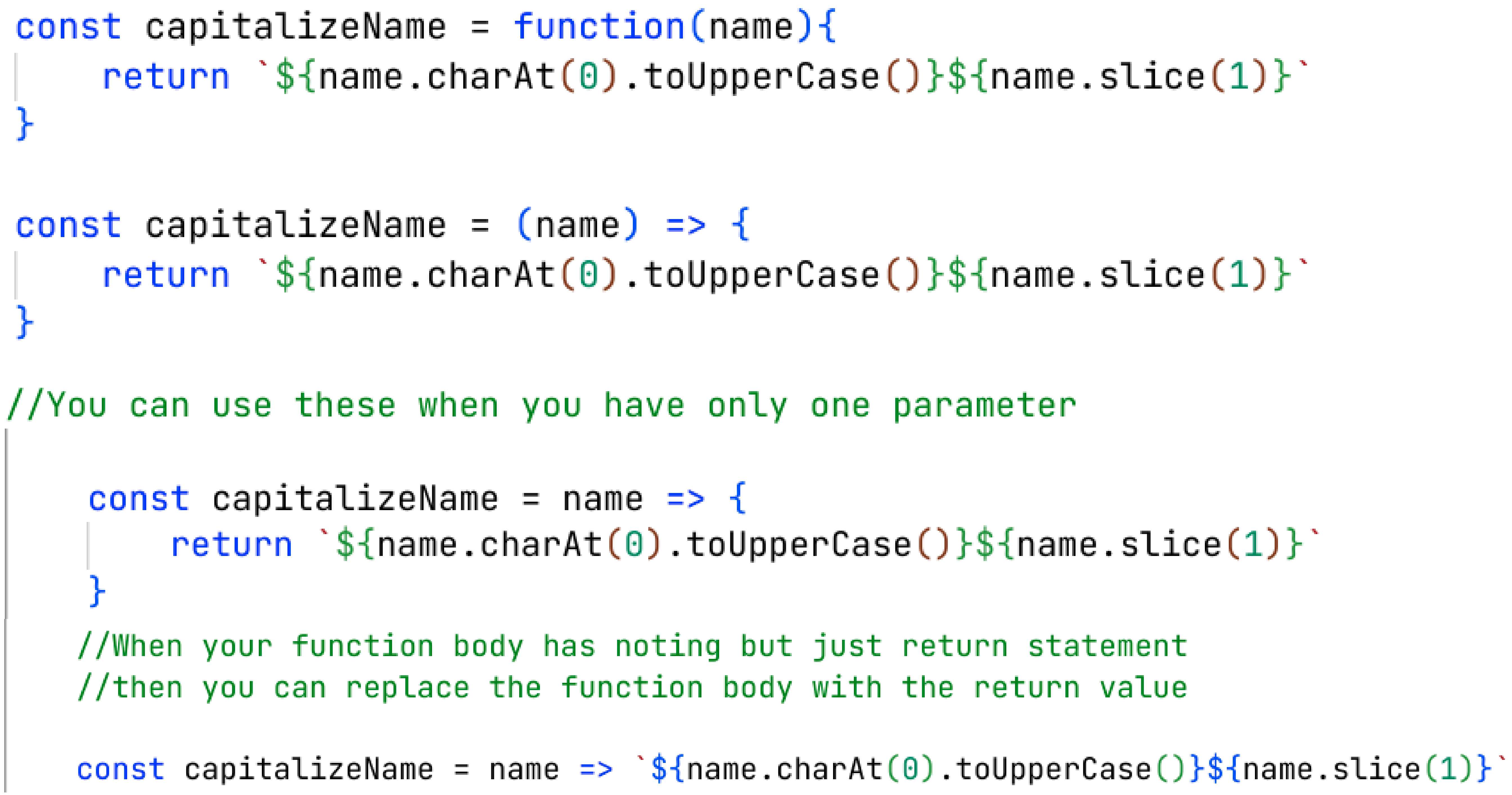
```
/**
 * General Expression
 *     do-while loop
 * initialization
 * do{
 *      body of the loop
 *      updation
 * }while(condition)
 */

let i = 1;
do{
    console.log(`Ashu-${i}`)
    i++
}while(i <= 10)
```

Functions



```
/**
 * General Expression
 *      function
 *
 * function fnName(input){
 *      body
 * }
 */

// Function Declaration
function capitalizeName(name){
    return `${name.charAt(0).toUpperCase()}${name.slice(1)}`
}

// Function call
console.log(capitalizeName('unga'))
```

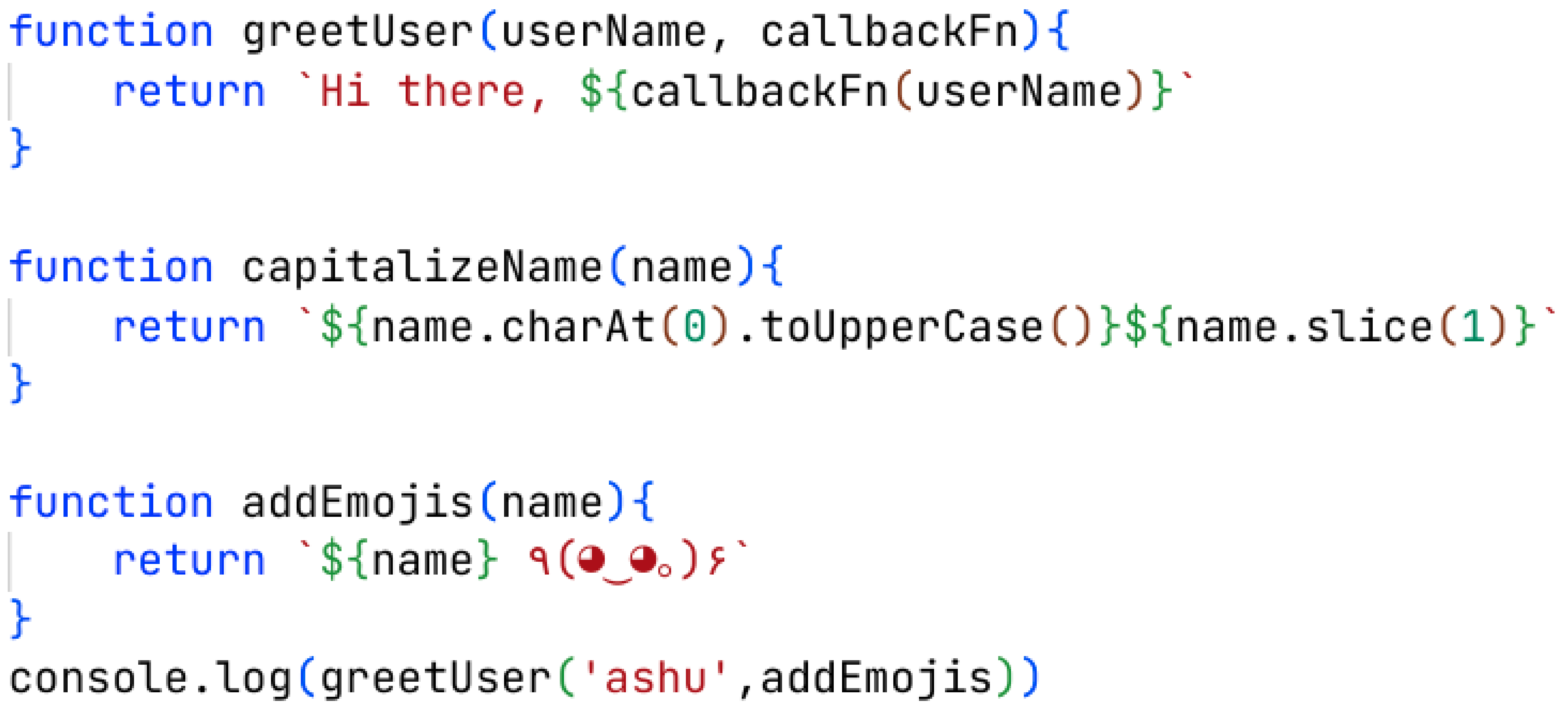
Arrow Functions



```
const capitalizeName = function(name){
    return `${name.charAt(0).toUpperCase()}${name.slice(1)}`
}

const capitalizeName = (name) => {
    return `${name.charAt(0).toUpperCase()}${name.slice(1)}`
}

//You can use these when you have only one parameter

    const capitalizeName = name => {
        return `${name.charAt(0).toUpperCase()}${name.slice(1)}`
    }
    //When your function body has noting but just return statement
    //then you can replace the function body with the return value

    const capitalizeName = name => `${name.charAt(0).toUpperCase()}${name.slice(1)}`
```

Callback Functions

A callback is a function passed as an argument to another function.

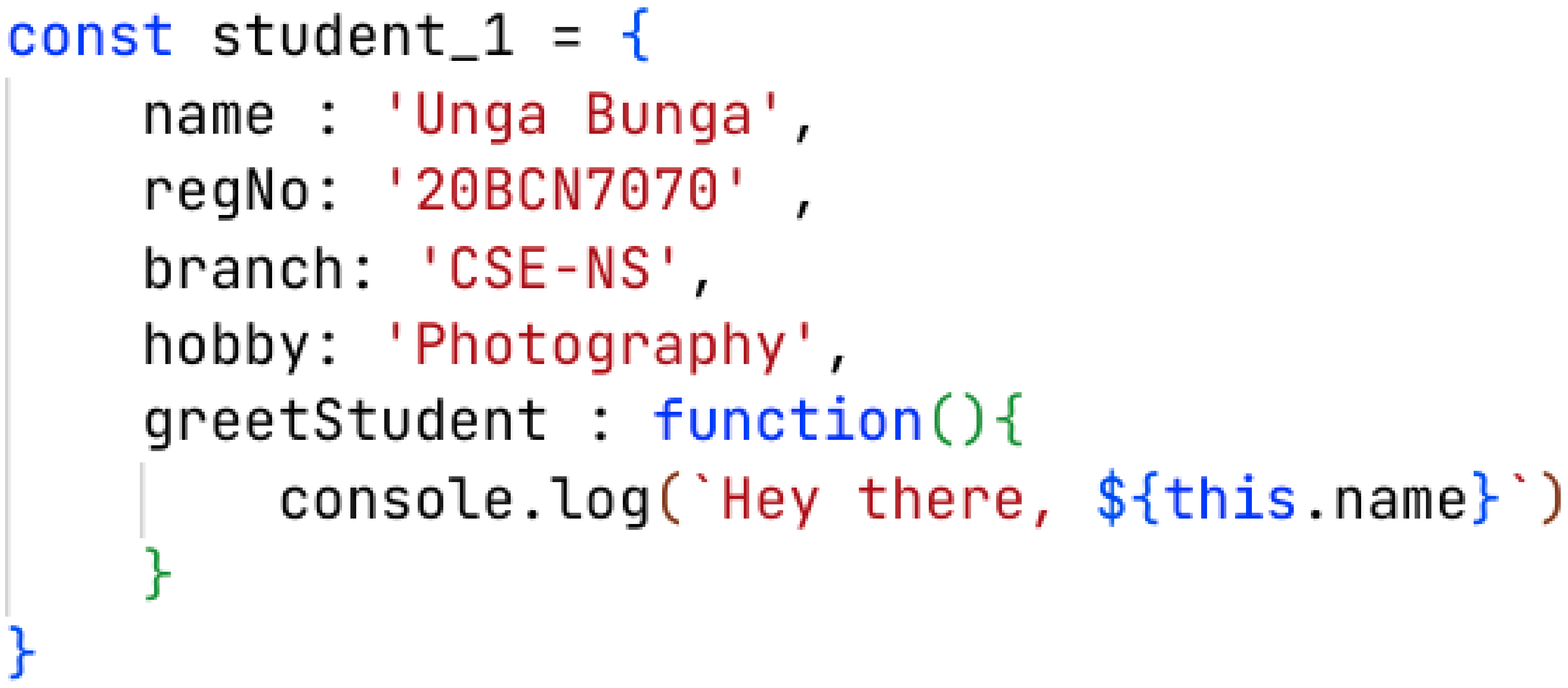


```
function greetUser(userName, callbackFn){
    return `Hi there, ${callbackFn(userName)}`
}

function capitalizeName(name){
    return `${name.charAt(0).toUpperCase()}${name.slice(1)}`
}

function addEmojis(name){
    return `${name} ٩(◕‿◕｡)۶`
}
console.log(greetUser('ashu',addEmojis))
```

Objects

Objects, in JavaScript, are the most important data type and form the building blocks for modern JavaScript. These objects are quite different from JavaScript’s primitive data types (Number, String, Boolean, null, undefined, and symbol) in the sense that these primitive data types all store a single value each (depending on their types)

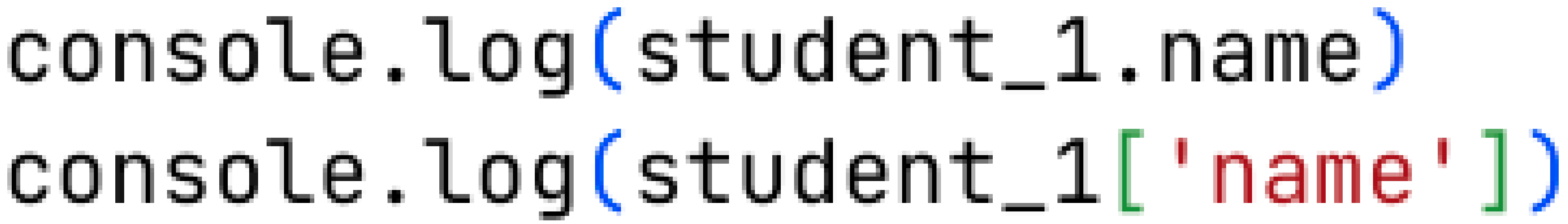
Defining an Object



```
const student_1 = {
    name : 'Unga Bunga',
    regNo: '20BCN7070' ,
    branch: 'CSE-NS',
    hobby: 'Photography',
    greetStudent : function(){
        console.log(`Hey there, ${this.name}`)
    }
}
```

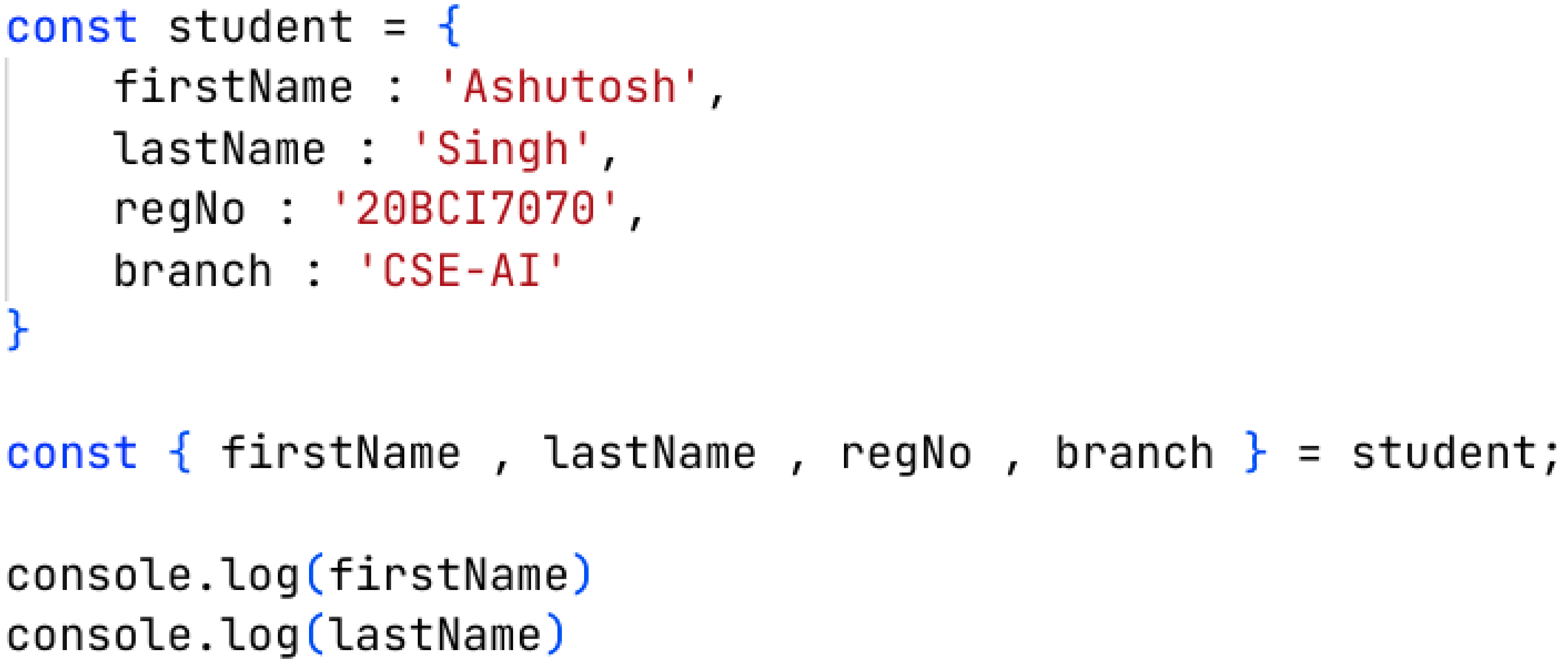
Accessing the values of an Object

1. Using the dot operator (.) Example : student_1.name
2. Using square brackets ([]) Example : student_1['name']



```
console.log(student_1.name)
console.log(student_1['name'])
```

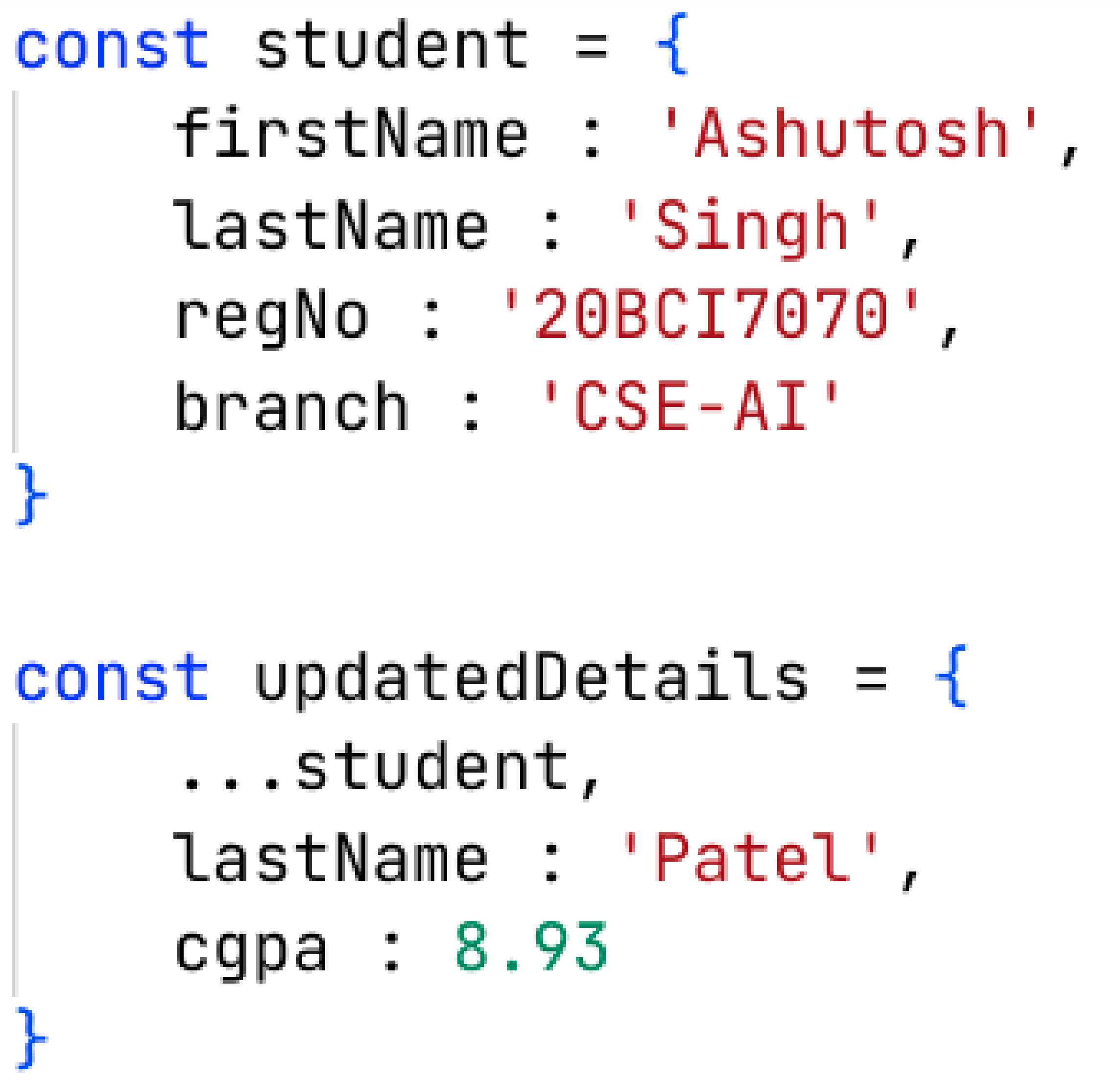
Object destructuring



```
const student = {
    firstName : 'Ashutosh',
    lastName : 'Singh',
    regNo : '20BCI7070',
    branch : 'CSE-AI'
}

const { firstName , lastName , regNo , branch } = student;

console.log(firstName)
console.log(lastName)
```

Spread Operator

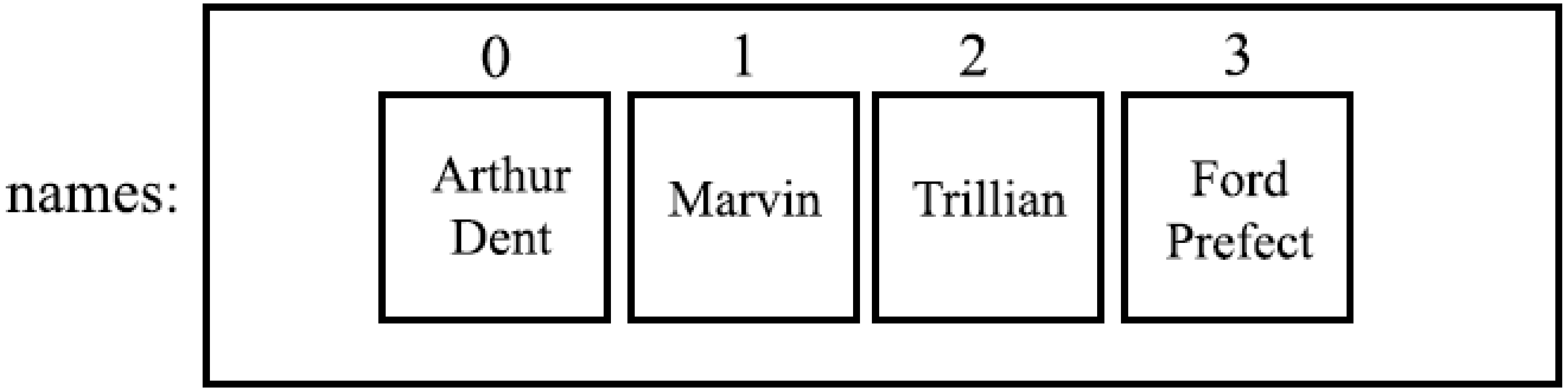
It can be used for creating new object with update values and more features.



```
const student = {
    firstName : 'Ashutosh',
    lastName : 'Singh',
    regNo : '20BCI7070',
    branch : 'CSE-AI'
}

const updatedDetails = {
    ...student,
    lastName : 'Patel',
    cgpa : 8.93
}
```

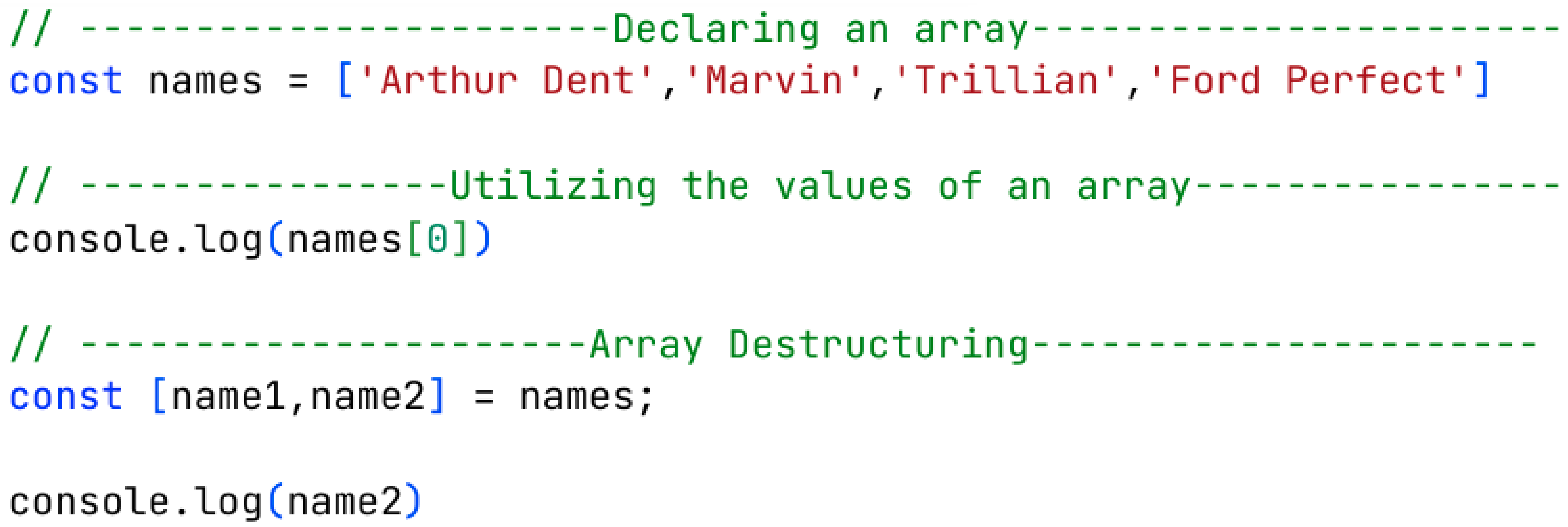
Array

JavaScript array is a single variable that is used to store multiple elements of the same type.



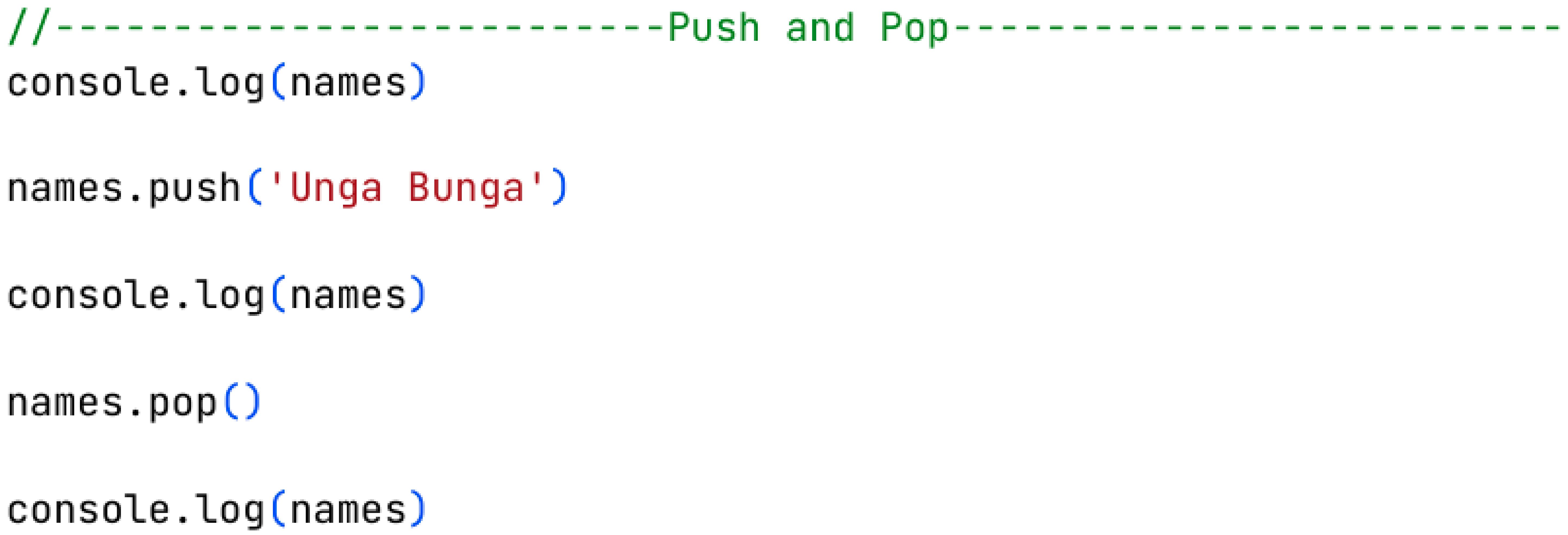
	0	1	2	3
names:	Arthur Dent	Marvin	Trillian	Ford Prefect

Array



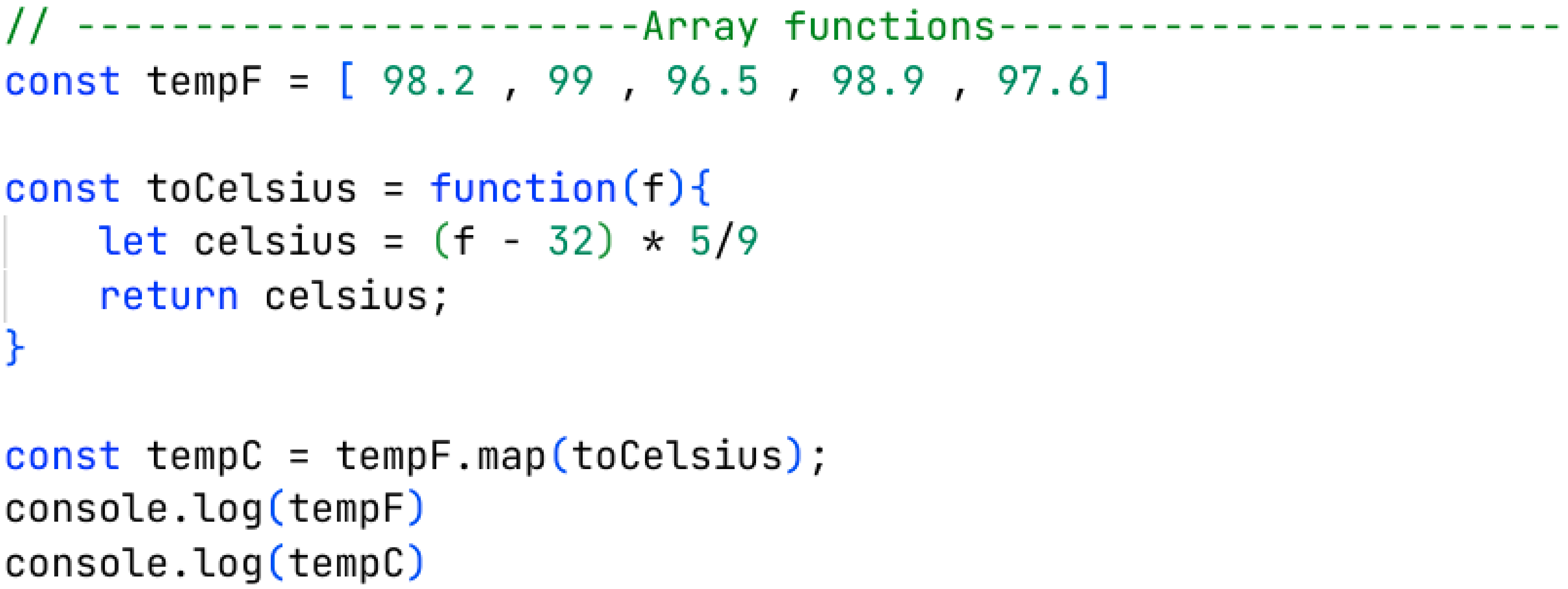
```
// -----------------------Declaring an array-----------------------
const names = ['Arthur Dent','Marvin','Trillian','Ford Perfect']

// ----------------Utilizing the values of an array----------------
console.log(names[0])

// ----------------------Array Destructuring----------------------
const [name1,name2] = names;

console.log(name2)
```

Push and Pop operations



```
//--------------------------Push and Pop--------------------------
console.log(names)

names.push('Unga Bunga')

console.log(names)

names.pop()

console.log(names)
```

array.prototype.map()



```
// ------------------------Array functions------------------------
const tempF = [ 98.2 , 99 , 96.5 , 98.9 , 97.6]

const toCelsius = function(f){
    let celsius = (f - 32) * 5/9
    return celsius;
}

const tempC = tempF.map(toCelsius);
console.log(tempF)
console.log(tempC)
```

Some other array functions

- filter()
- reduce()
- some() / every()
- find() / findIndex()
- forEach()
- slice()
- concat()
- includes()

Array - JavaScript | MDN

The Array object, as with arrays in other programming languages, enables storing a collection of multiple items under a single variable name, and has members for performing common array operations.

mozilla.org

Thank You

If you have any doubt or queries you can contact me on my socials

- LinkedIn
- Discord
- Email