CADT IDT
Next-Gen Engagement Program

JAVASCRIPT

Week 1

Introduction to JavaScript + Converting C to JS

Ory Chanraksa & Sao Visal

LESSON OVERVIEW

Welcome to your first week of diving into JavaScript! This lesson introduces the basics of JavaScript, including:

- Intro to Javascript
- Printing
- Variables
- Data types
- Operator
- Control Statements
- Loops
- Objects
- Arrays
- Converting C to JavaScript

TOOLS & EXTENSIONS

VS Code – Fast, smart editor with debugging, Git, and JavaScript support.

Chrome – Powerful and versatile DevTools for debugging, inspecting, and optimizing web apps efficiently.

Prettier - Automatically formats your code to ensure consistent style and readability

Error lens - Detect error inside the code

Path Intellisense - Auto complete files name, useful for importing files

NPM Intellisense - Auto-completes npm module names in the import statements to save time and avoid typos

Live Server - Launch a development local Server with live reload feature for static & dynamic pages.

vscode-pdf - Display pdf file in VSCode.

Link to full installation guide: https://www.canva.com/design/DAGu6U3m6Mw/xpYZHAK6o2N-Ni-oDKCuEA/edit?utm_content=DAGu6U3m6Mw&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

WHAT IS JAVASCRIPT?

- JavaScript is a powerful, high-level programming language commonly used to create interactive effects on websites. It’s one of the core technologies of the web, alongside HTML and CSS.
- JavaScript is interpreted, dynamically typed, and loosely typed, which means you don’t need to explicitly define data types like in C.

WHAT IS JAVASCRIPT?

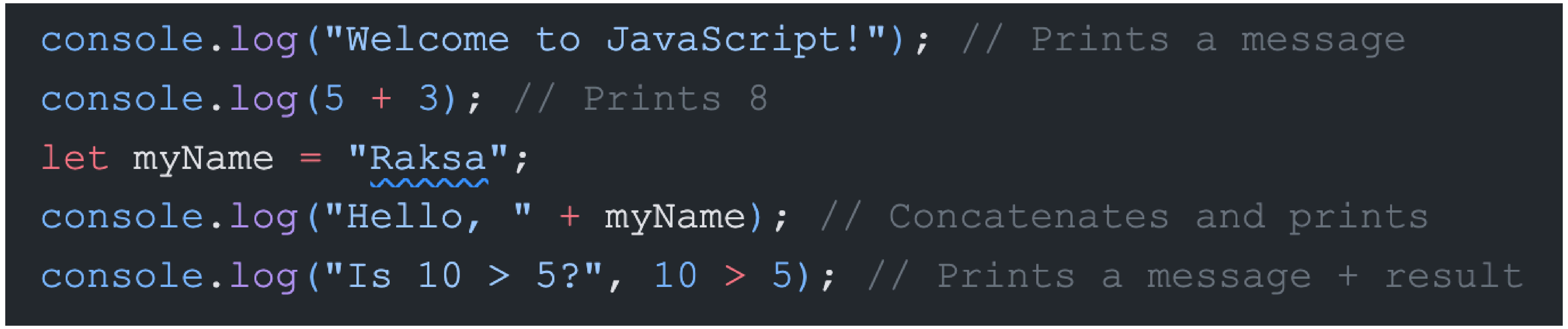
- Created in 1995, originally meant for web browsers.
- Now used on the front-end and back-end (thanks to Node.js).
- Runs directly in the browser without the need for a compiler.
- Used to manipulate web pages (DOM), make network requests, handle user input, and more.

PRINTING

In JavaScript, printing means sending output to the browser's developer console, or terminal.

This is done using the console.log() function, which is mainly used for:

- Displaying values
- Debugging code
- Checking flow and output during development



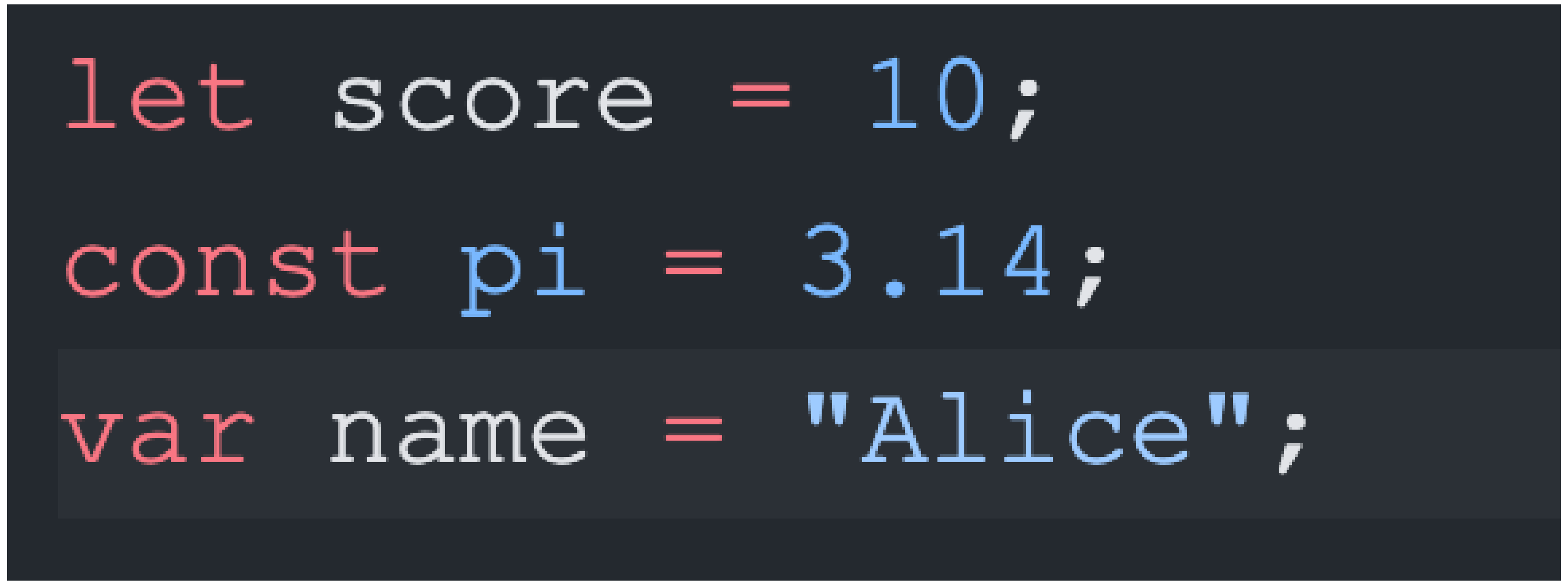
```
console.log("Welcome to JavaScript!"); // Prints a message
console.log(5 + 3); // Prints 8
let myName = "Raksa";
console.log("Hello, " + myName); // Concatenates and prints
console.log("Is 10 > 5?", 10 > 5); // Prints a message + result
```

Unlike in C (where we use printf()), JavaScript doesn't need format specifiers like %d or %s.

VARIABLES

Variables are used to store and manage data in your programs.

- **var** — function-scoped (older, mostly avoided today)
- **let** — block-scoped, reassignable
- **const** — block-scoped, cannot be reassigned



```
let score = 10;
const pi = 3.14;
var name = "Alice";
```

*JavaScript **doesn't require** you to declare a type like int, float, or char—you just assign a value.*

DATA TYPES

JavaScript includes both primitive and non-primitive (reference) data types.

Primitive Types:

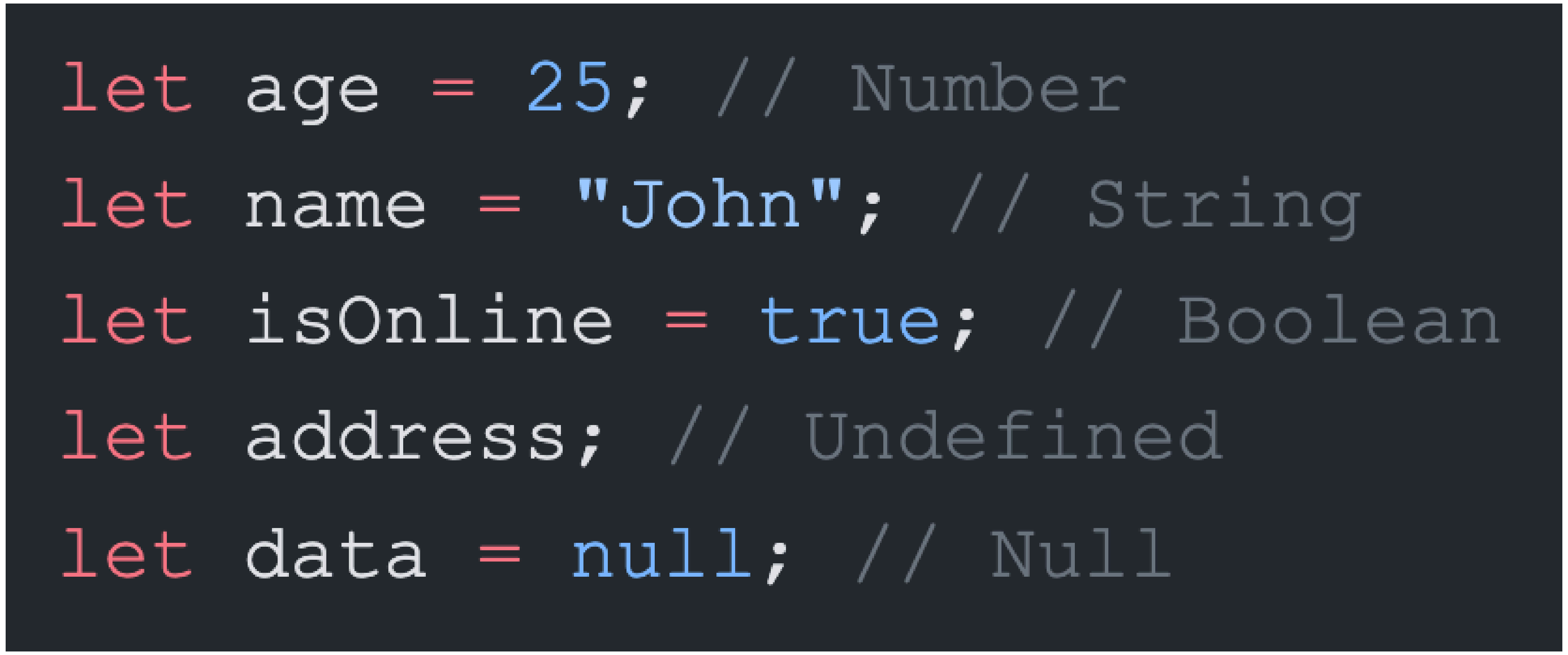
- **Number** – for all numeric values (no separate int or float)
- **String** – text, wrapped in " " or ' '
- **Boolean** – true or false
- **Undefined** – variable declared but not assigned
- **Null** – explicitly no value

DATA TYPES

- **BigInt** – for large integers (used rarely)
- **Symbol** – unique and immutable (used in advanced cases)

Non-Primitive Types:

- Object
- Array
- Function



```
let age = 25; // Number
let name = "John"; // String
let isOnline = true; // Boolean
let address; // Undefined
let data = null; // Null
```

*JavaScript is **dynamically typed**, so the type of a variable can change during runtime.*

OPERATORS

Operators are used to perform actions on variables and values.

Arithmetic Operators:

- +, -, *, /, %, ** (exponentiation)

Assignment Operators:

- =, +=, -=, *=, /=, %=

Comparison Operators:

- == equal (loose equality, type conversion allowed)

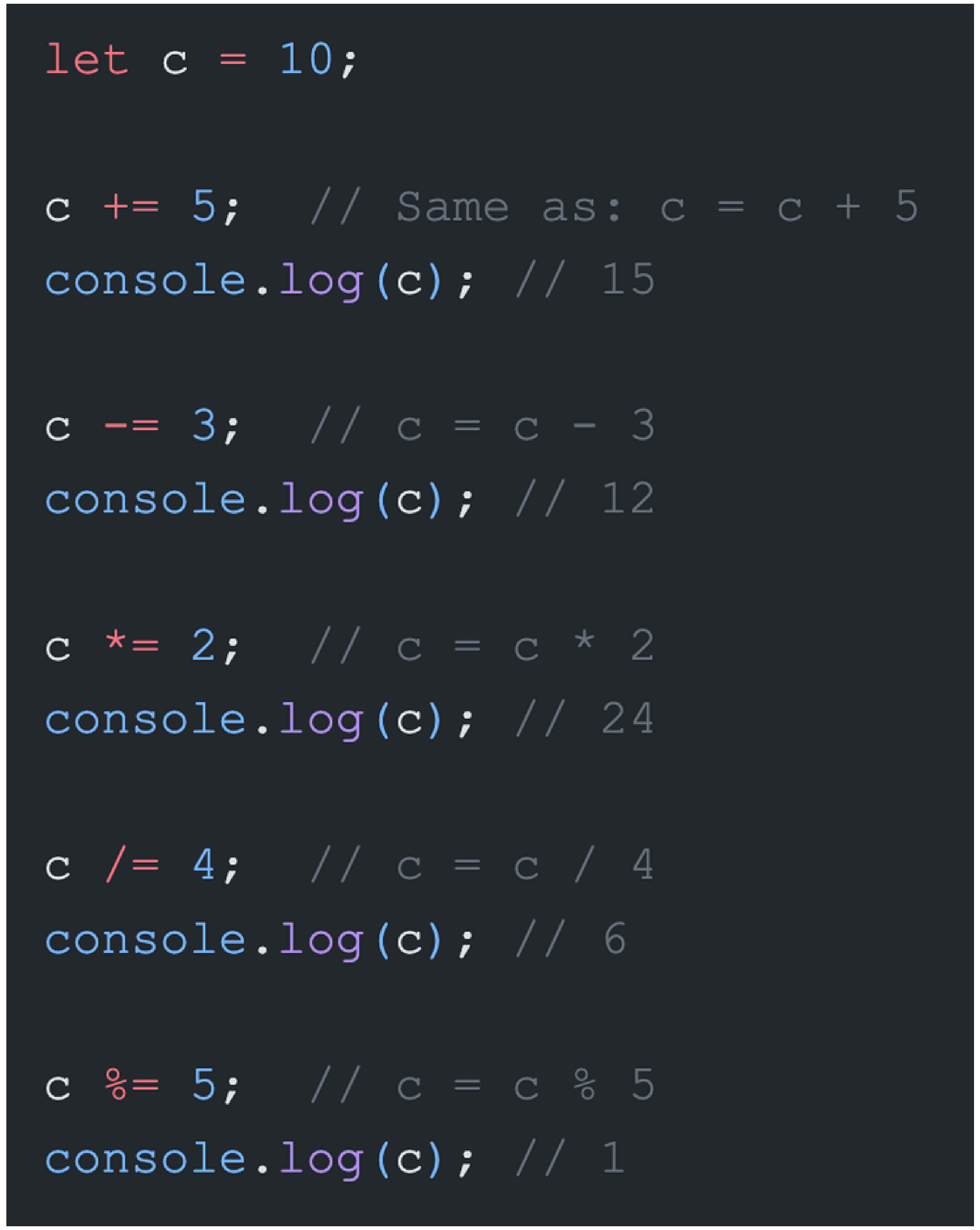
OPERATORS

- === strict equal (type + value must match)
- !=, !==, >, <, >=, <=

Logical Operators:

- && logical AND
- | | logical OR
- ! logical NOT

```javascript
// Arithmetic
let a = 4;
let b = 2;

console.log(a ** b);  // 16 (Exponentiation: 4^2)
console.log(a % b);   // 0  (Remainder)
```



```
let c = 10;

c += 5;  // Same as: c = c + 5
console.log(c); // 15

c -= 3;  // c = c - 3
console.log(c); // 12

c *= 2;  // c = c * 2
console.log(c); // 24

c /= 4;  // c = c / 4
console.log(c); // 6

c %= 5;  // c = c % 5
console.log(c); // 1
```

```
// Comparison
let score = 90;
let grade = "90";

console.log(score == grade);   // true  (value is the same)
console.log(score === grade);  // false (type is different)

console.log(score > 80);       // true
console.log(score <= 100);     // true

console.log(score != 50);      // true
console.log(score !== "90");   // true
```

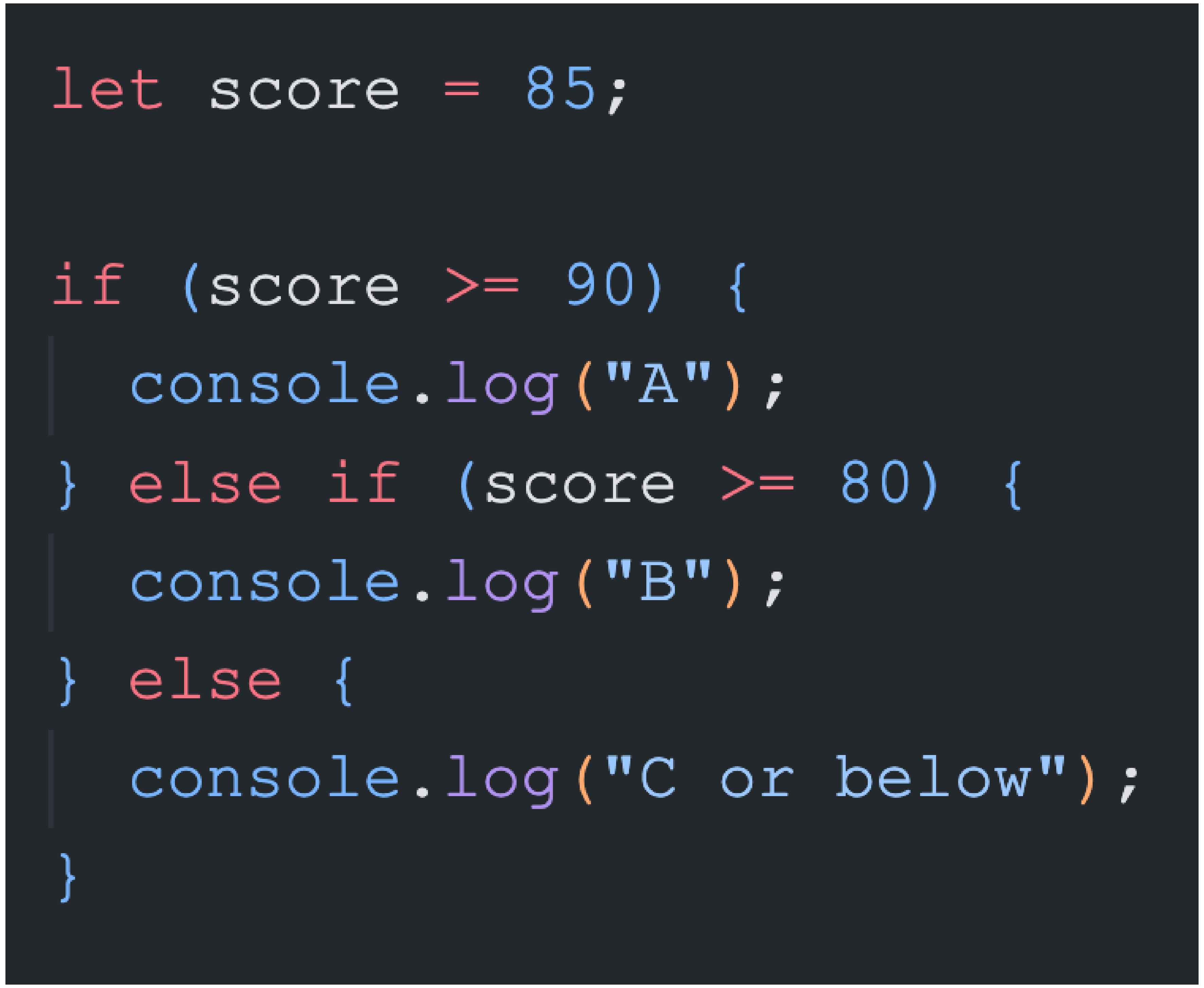
```
// Logical
let isLoggedIn = true;
let isAdmin = false;

console.log(isLoggedIn && isAdmin); // false (must be both true)
console.log(isLoggedIn || isAdmin); // true  (at least one is true)
console.log(!isAdmin);              // true  (not false)
```

CONTROL STATEMENT

Control statements are used to control the flow of the program — deciding which code runs or when to stop based on a condition.

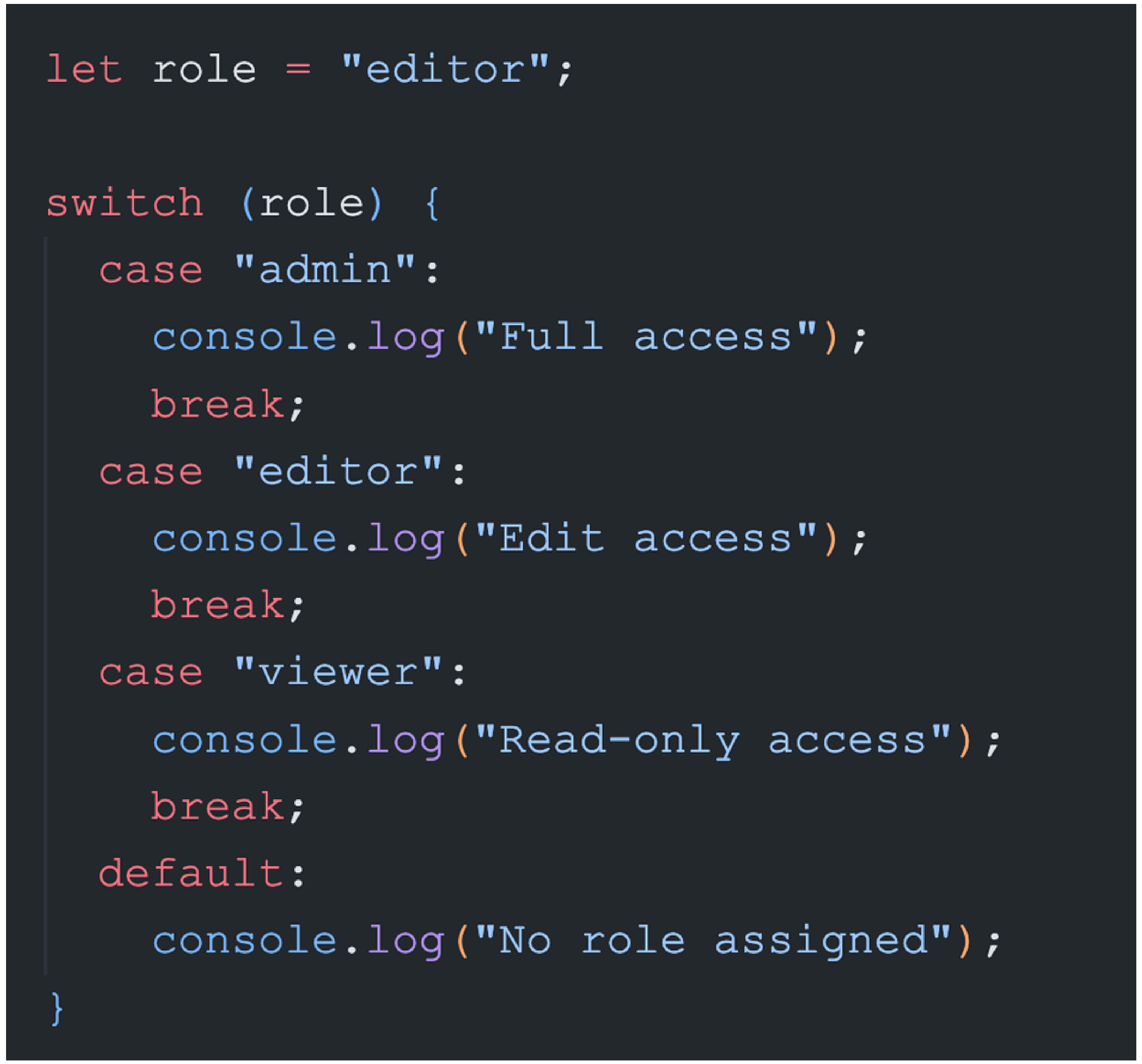
Control flow statements includes:

- if
- else if
- else
- switch



```
let score = 85;

if (score >= 90) {
  console.log("A");
} else if (score >= 80) {
  console.log("B");
} else {
  console.log("C or below");
}
```

Checks from top to bottom. ***First true block runs.***



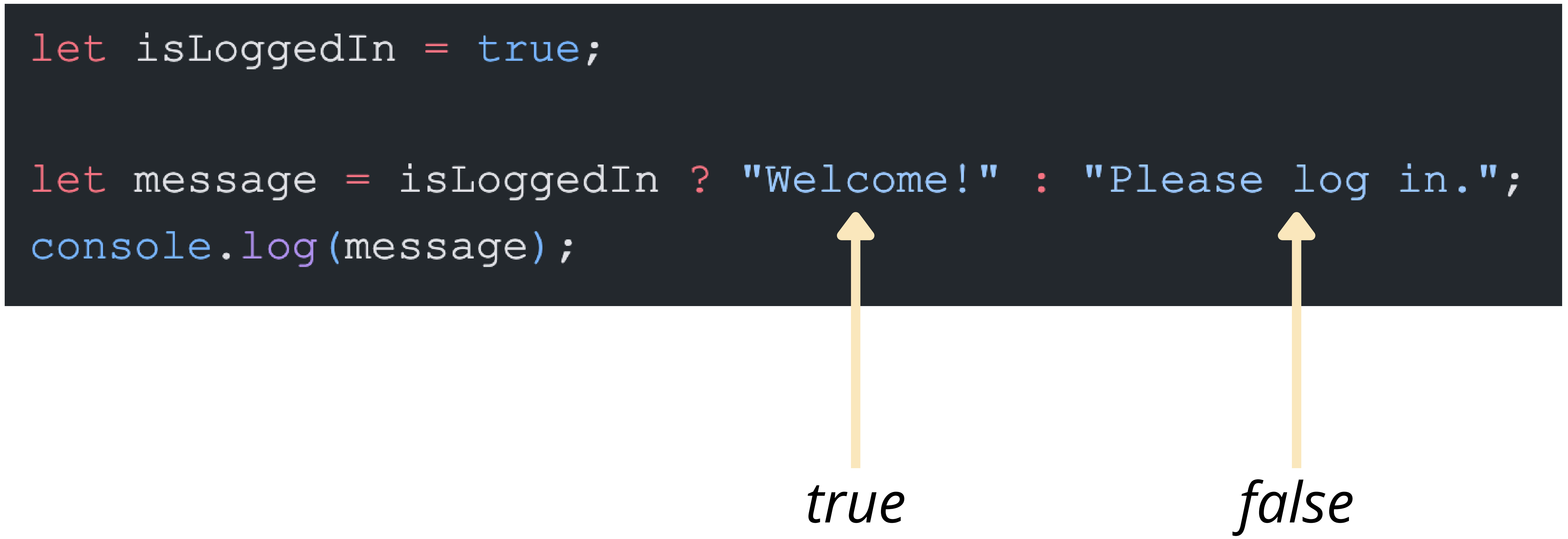
```javascript
let role = "editor";

switch (role) {
  case "admin":
    console.log("Full access");
    break;
  case "editor":
    console.log("Edit access");
    break;
  case "viewer":
    console.log("Read-only access");
    break;
  default:
    console.log("No role assigned");
}
```

- ***break*** *stops the case from continuing to the next one.*
- ***default*** *runs if no match is found.*
- *Good for checking fixed, known options like user roles, status codes, etc.*

CONTROL STATEMENT

Ternary Operator – A shorter version of if-else. Good for quick conditions.

Structure: condition ? trueValue : falseValue



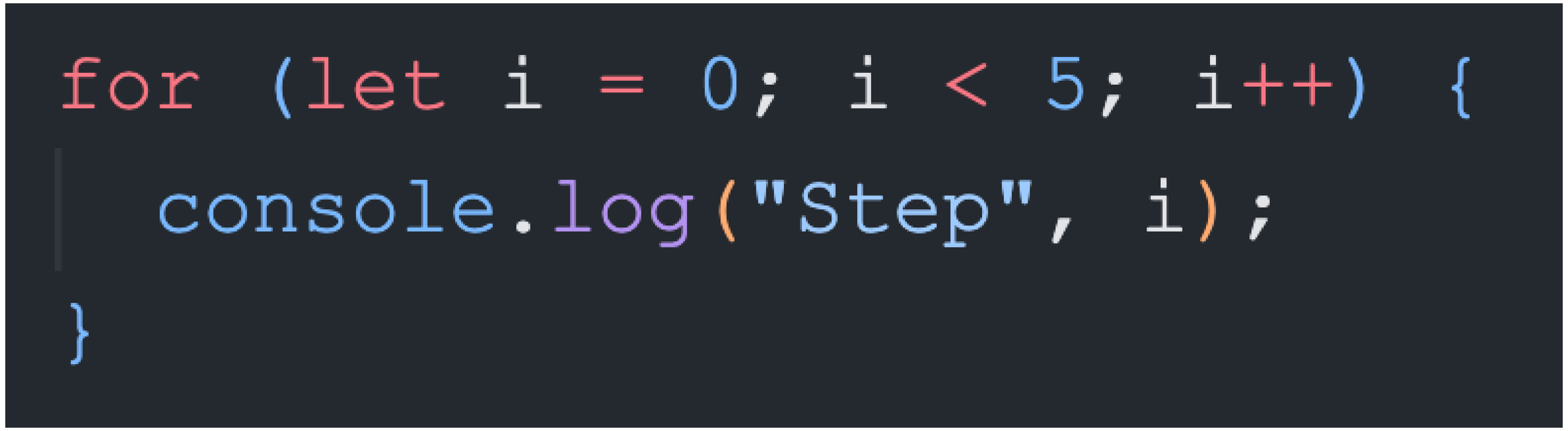
```
let isLoggedIn = true;

let message = isLoggedIn ? "Welcome!" : "Please log in.";
console.log(message);
```

LOOPS

Loops let you repeat code without rewriting it. Useful for arrays, repeating actions, and checking multiple values.

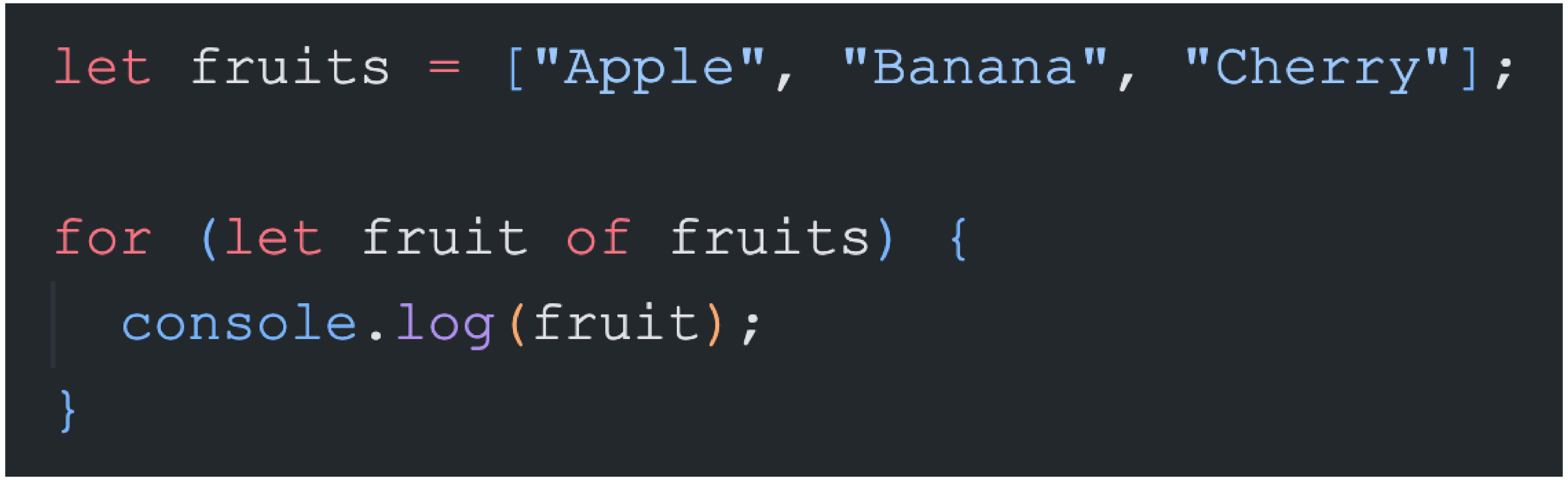
Types of loops in javascript includes:

- for
- for...of
- for...in
- forEach()
- while
- do...while

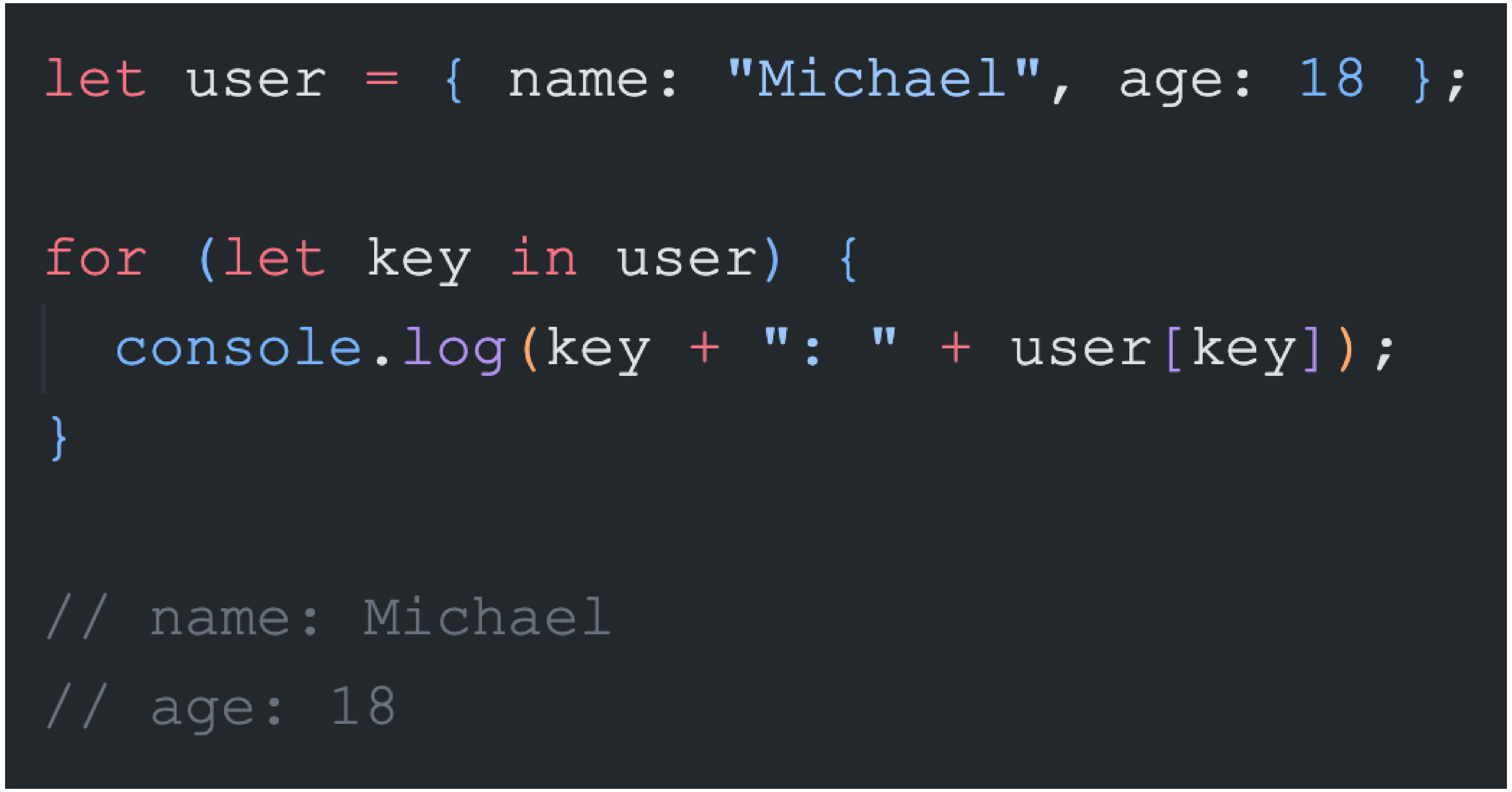


```
for (let i = 0; i < 5; i++) {
    console.log("Step", i);
}
```

*Use when you **know how many times** to loop.*


```
let fruits = ["Apple", "Banana", "Cherry"];

for (let fruit of fruits) {
  console.log(fruit);
}
```

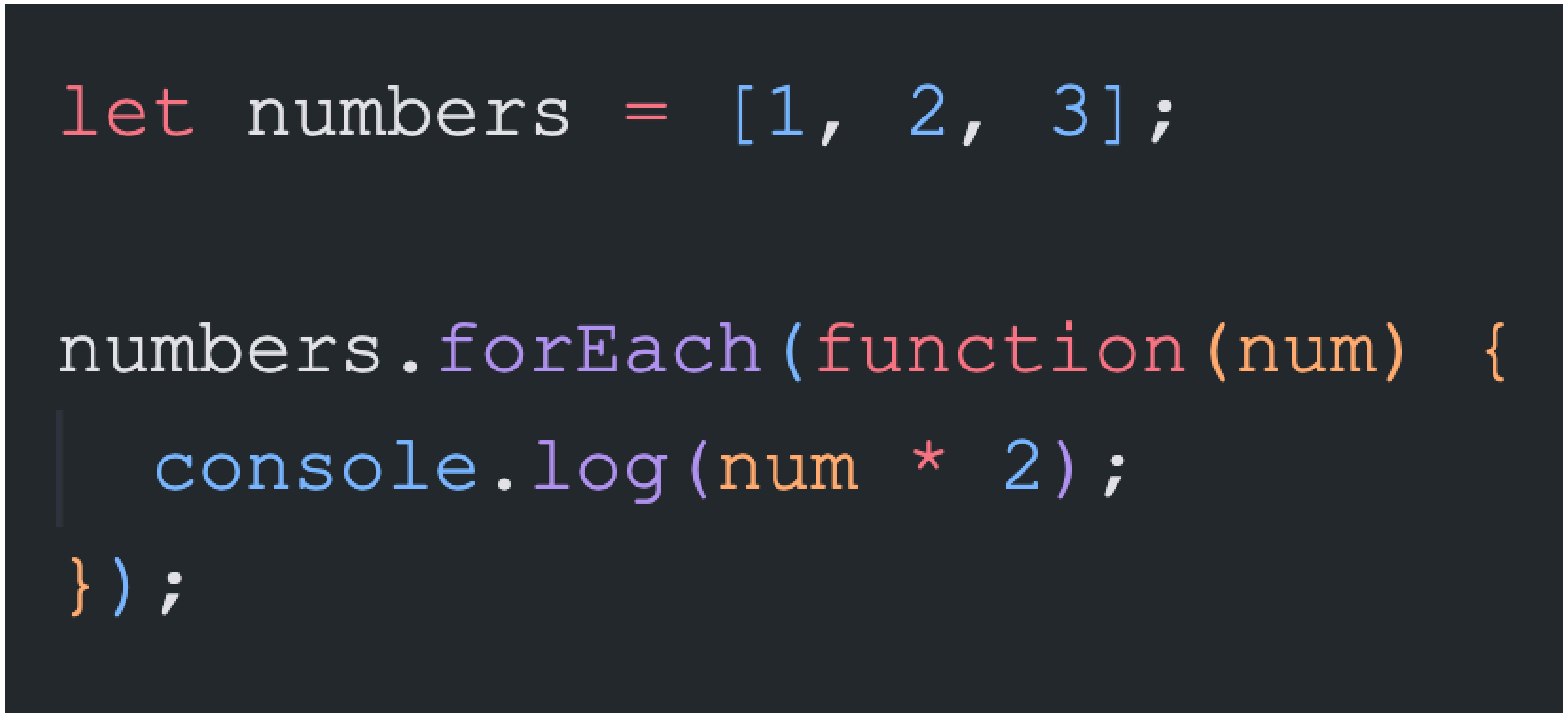
Loops through values in arrays or iterable objects.
Directly gives values (not index).



```
let user = { name: "Michael", age: 18 };

for (let key in user) {
  console.log(key + ": " + user[key]);
}


// name: Michael
// age: 18
```

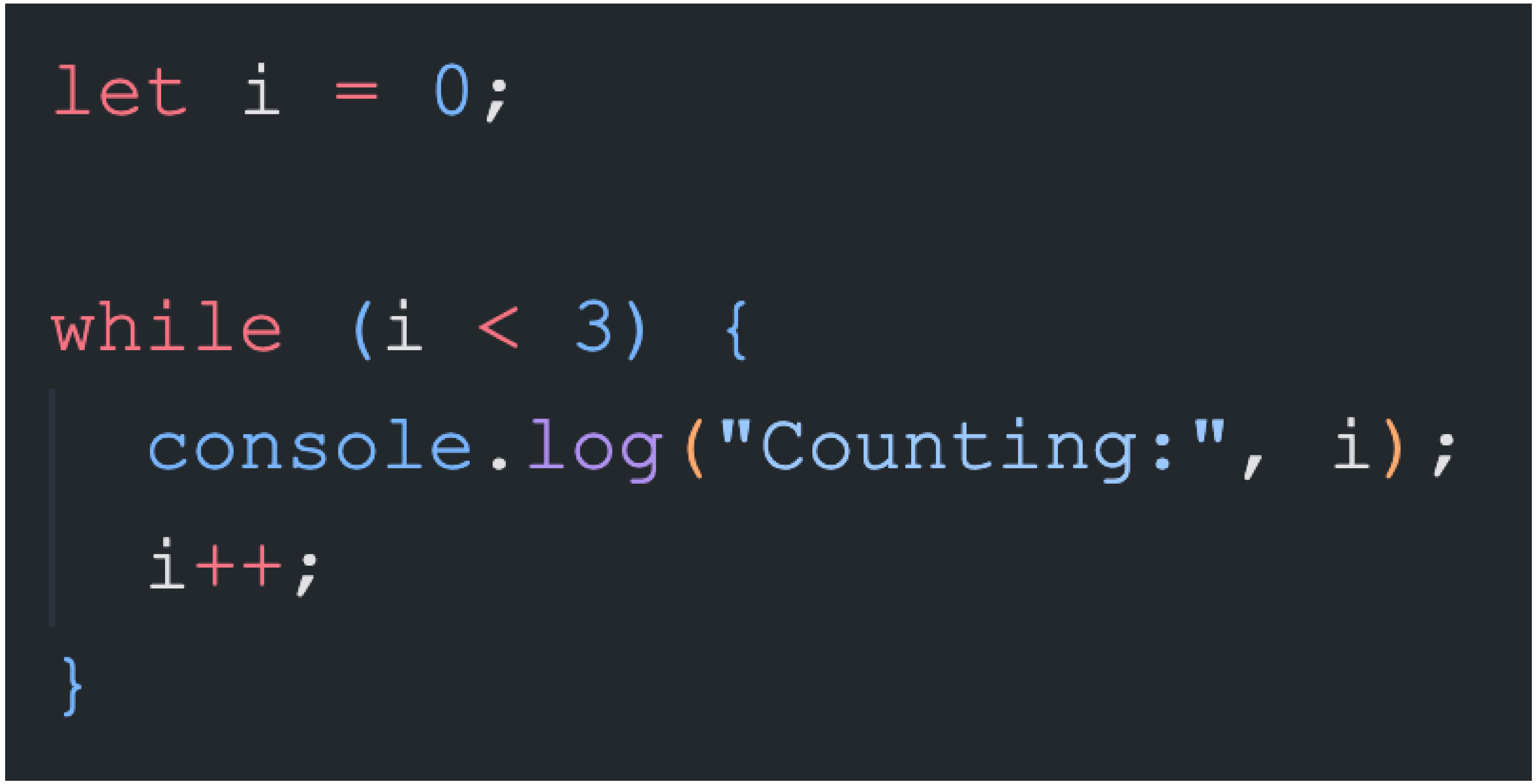
Loops through keys (property names) of an object.

Use for objects, not arrays.



```
let numbers = [1, 2, 3];

numbers.forEach(function(num) {
  console.log(num * 2);
});
```

*Loops through array elements using a function. Cleaner than a loop, but doesn't support **break** or **continue**.*



```
let i = 0;

while (i < 3) {
  console.log("Counting:", i);
  i++;
}
```

Repeats code as long as the condition is true.

Checks condition before each loop.

```
let j = 0;

do {
  console.log("This runs at least once:", j);
  j++;
} while (j < 3);
```

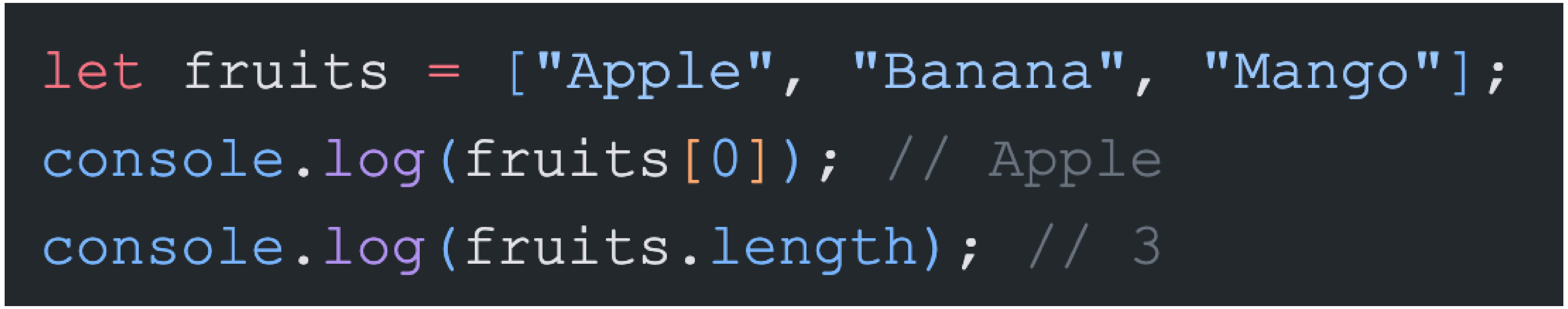
Always runs at least once, then checks the condition. ***Use when the code must run at least once, even if the condition is false initially.***

ARRAY

Arrays are ordered collections of values. They can hold any type: strings, numbers, booleans, or even objects and other arrays.

Unlike C, arrays in JavaScript:

- Don’t require a fixed size
- Can mix data types
- Offer many built-in methods



```
let fruits = ["Apple", "Banana", "Mango"];
console.log(fruits[0]); // Apple
console.log(fruits.length); // 3
```

ARRAY

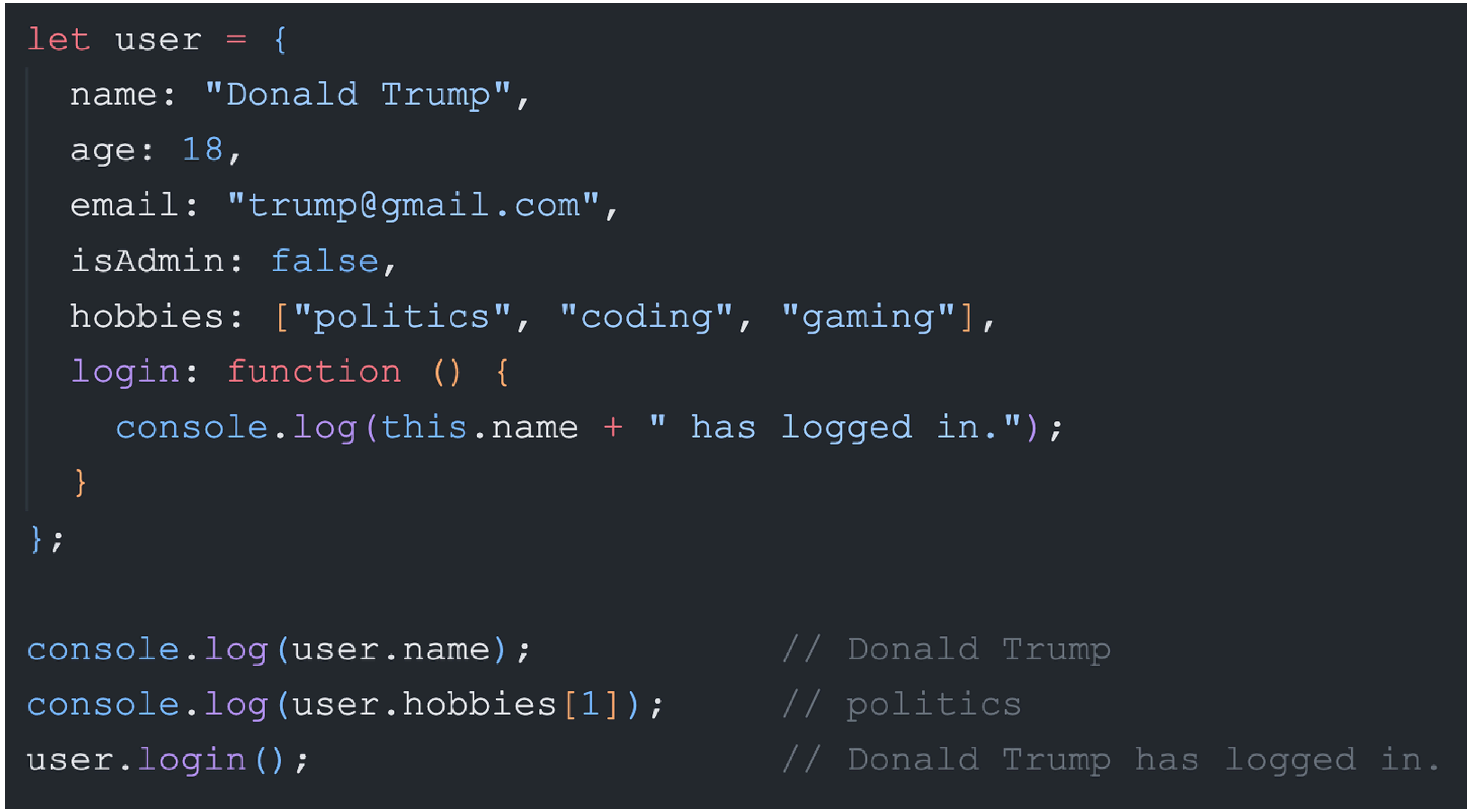
Common Methods:

- .push() – add to end
- .pop() – remove from end
- .shift() – remove from start
- .unshift() – add to start
- .forEach() – iterate
- .map() – create a new array from transformation
- .filter() - get elements based on condition(s)

OBJECT

Objects are collections of properties, written as **key-value pairs**.

Think of them like **structs** in C—but way more flexible. You can store any type of value inside an object: numbers, strings, arrays, functions, even other objects.



```javascript
let user = {
  name: "Donald Trump",
  age: 18,
  email: "trump@gmail.com",
  isAdmin: false,
  hobbies: ["politics", "coding", "gaming"],
  login: function () {
    console.log(this.name + " has logged in.");
  }
};

console.log(user.name);              // Donald Trump
console.log(user.hobbies[1]);        // politics
user.login();                        // Donald Trump has logged in.
```

- **this** refers to the object itself (used inside methods).
- You can access properties with dot notation **(user.name)** or bracket notation **(user["name"])**.
- You can add or change properties anytime.

CONVERT C TO JS

If you've written code in C before, many of the logic structures—if, while, for—exist in JavaScript too, but with different syntax and behavior.

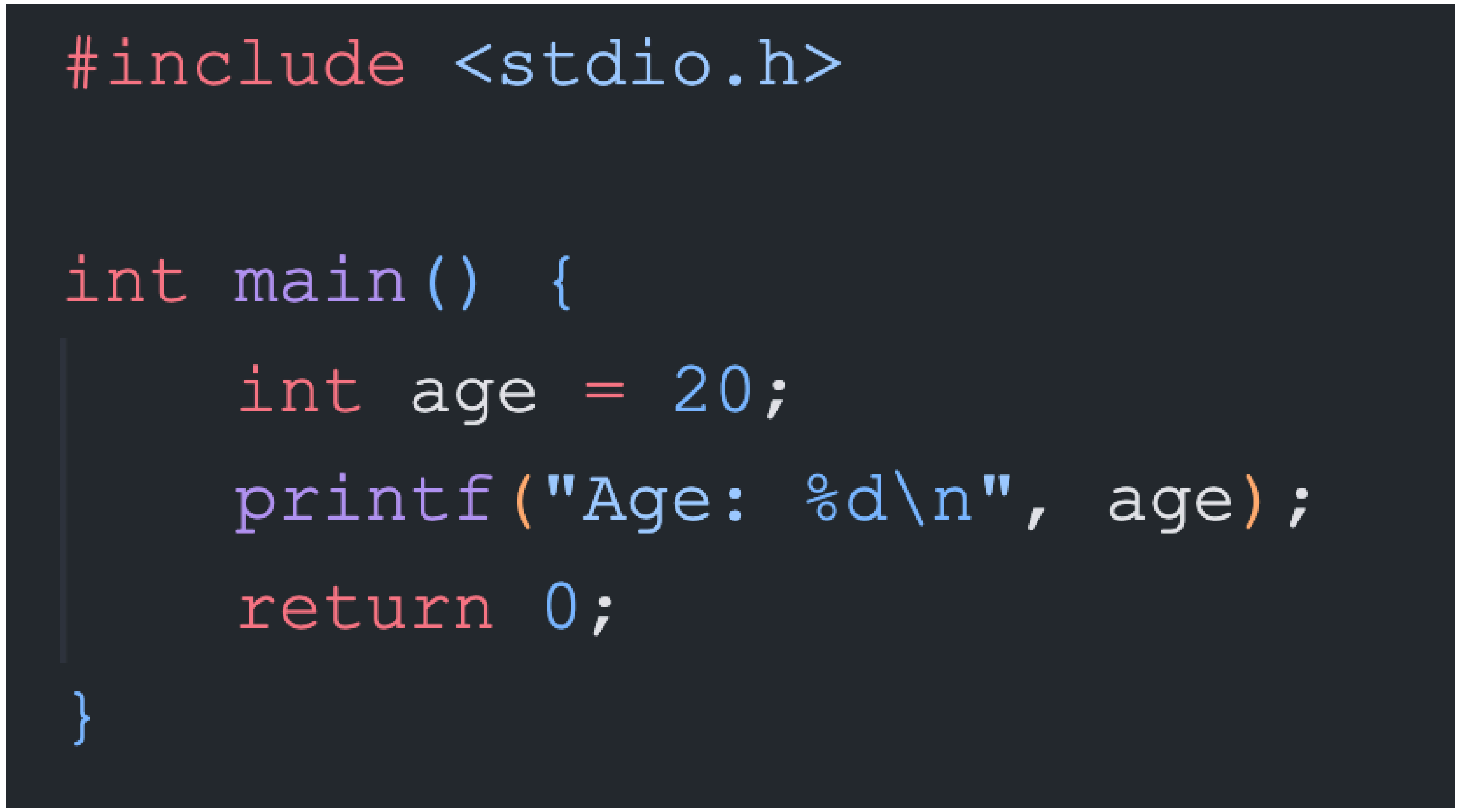
Key Differences:

- No need for header files or main()
- Variables don't need data types (no int, float, etc.)
- Use console.log() instead of printf()

CONVERT C TO JS

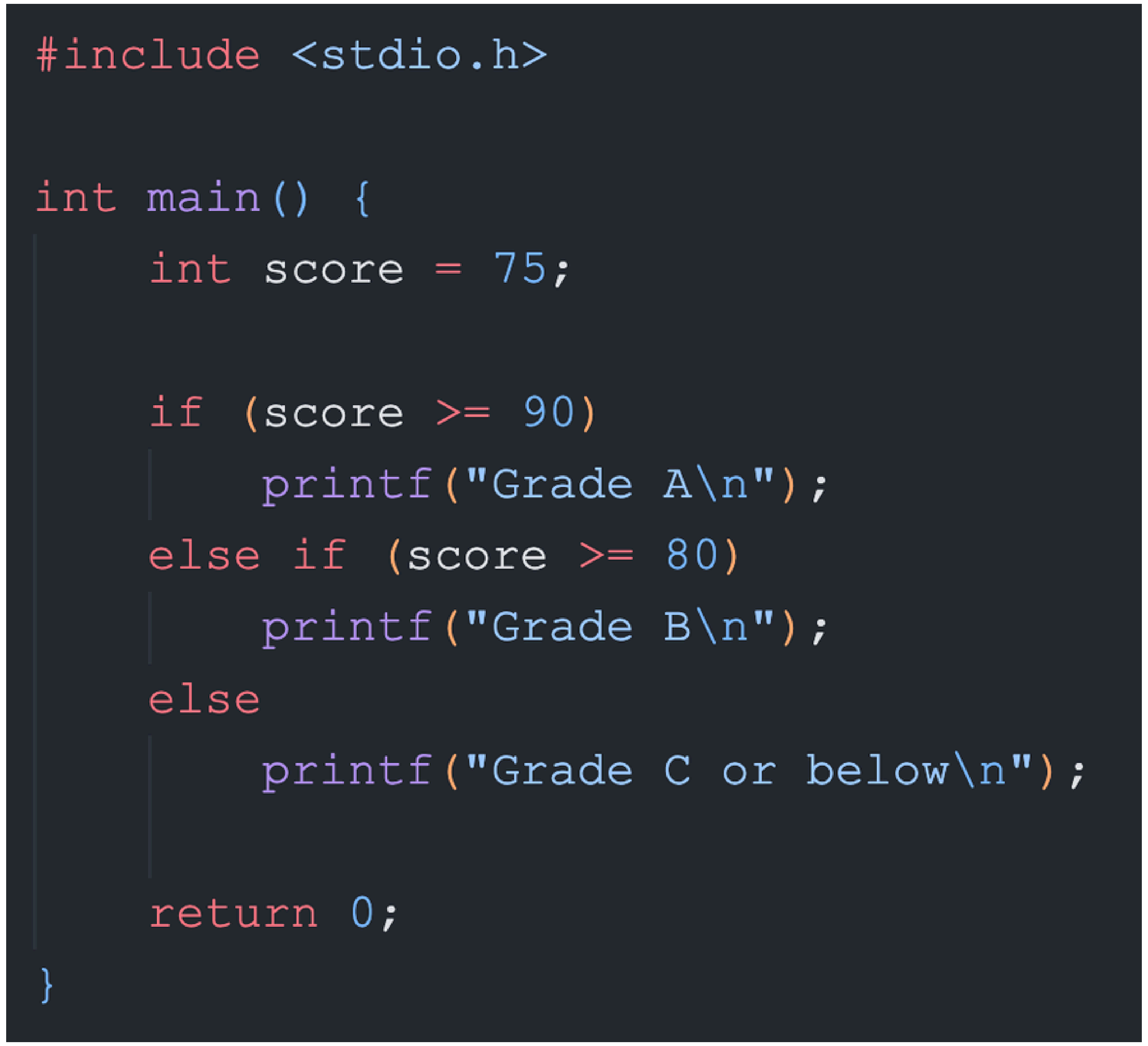
- No semicolon enforcement (though recommended)
- Memory management is automatic (no malloc/free)

Exercise 1



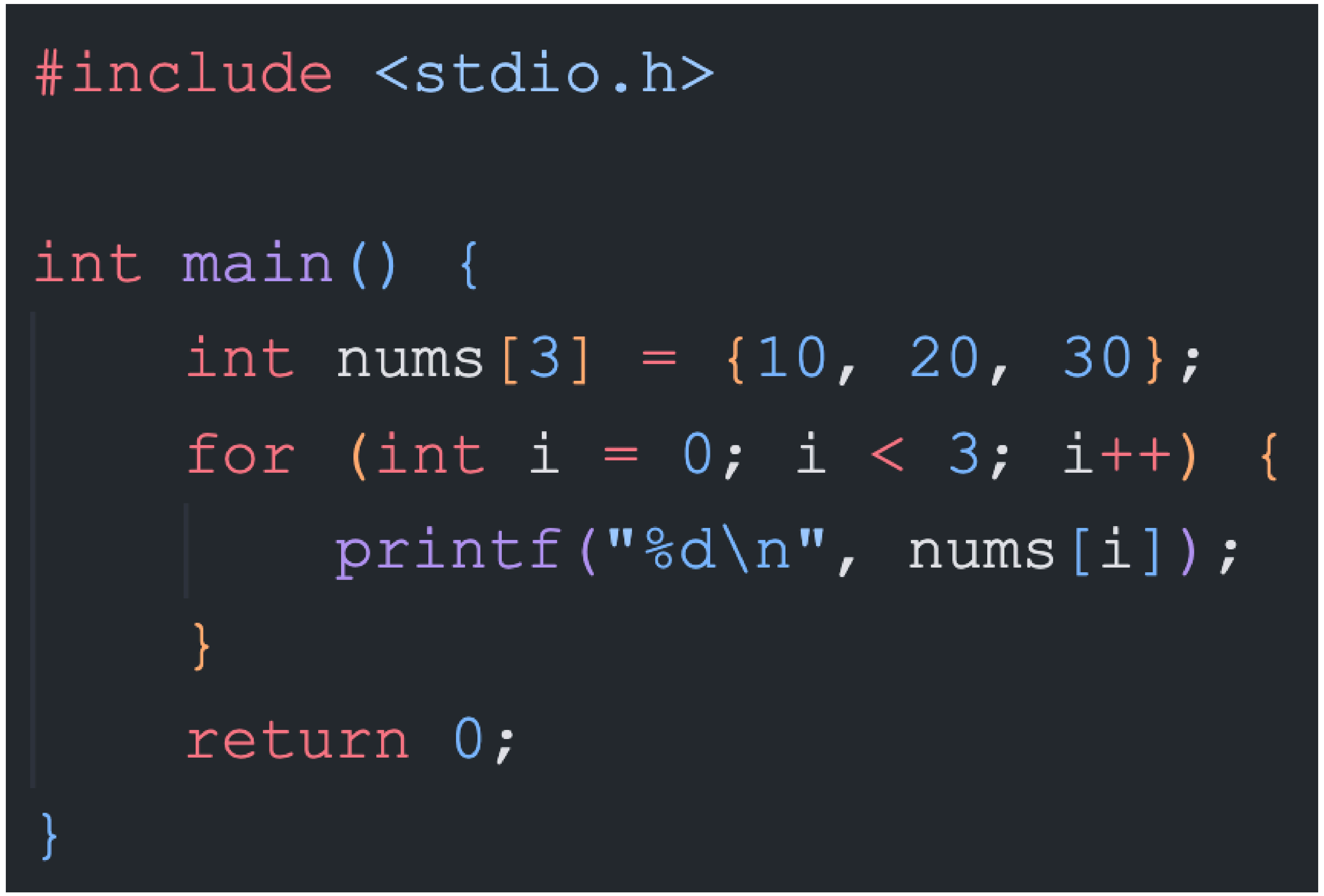
```c
#include <stdio.h>

int main() {
    int age = 20;
    printf("Age: %d\n", age);
    return 0;
}
```

Exercise 2



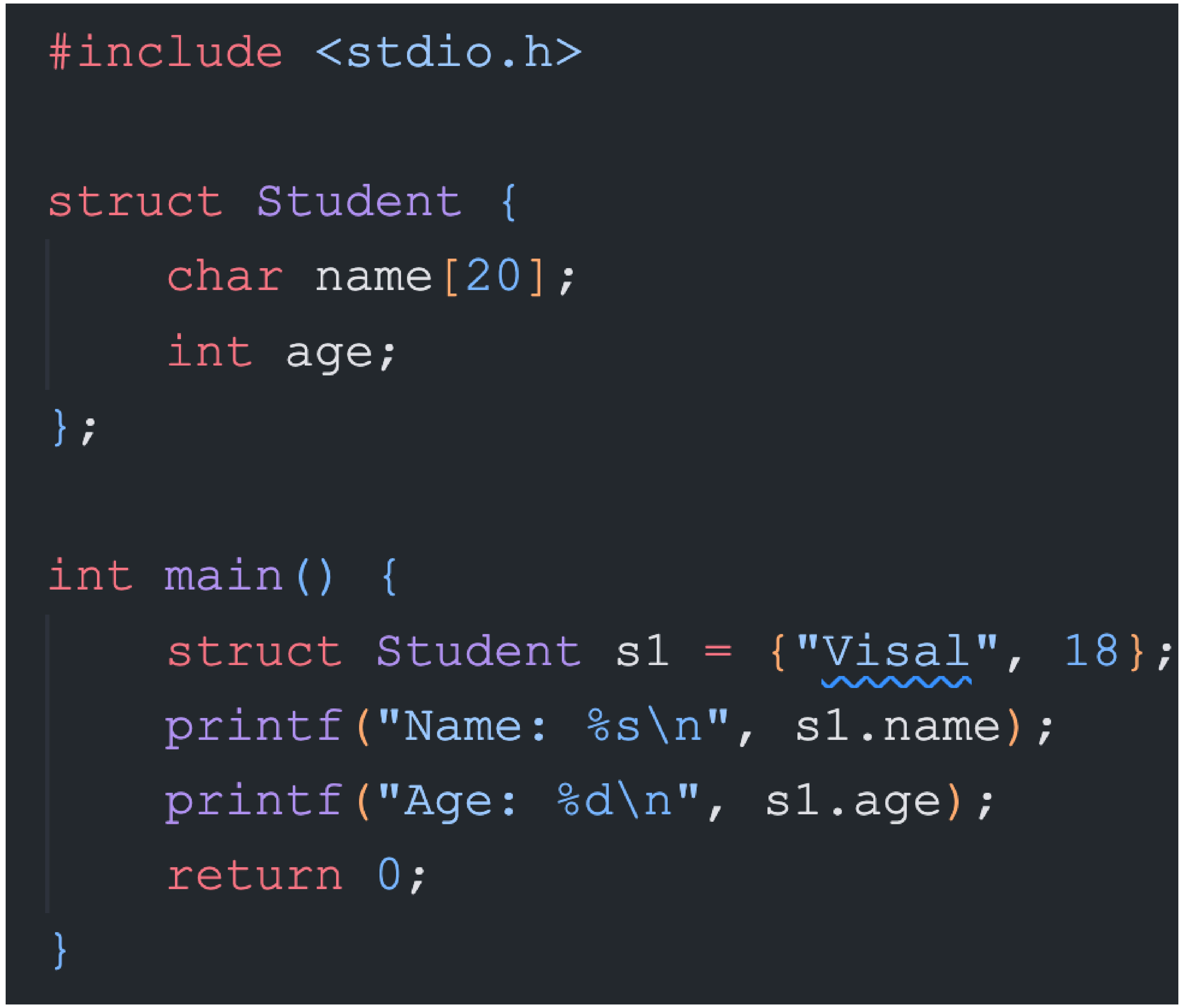
```c
#include <stdio.h>

int main() {
    int score = 75;

    if (score >= 90)
        printf("Grade A\n");
    else if (score >= 80)
        printf("Grade B\n");
    else
        printf("Grade C or below\n");

    return 0;
}
```

Exercise 3



```c
#include <stdio.h>

int main() {
    int nums[3] = {10, 20, 30};
    for (int i = 0; i < 3; i++) {
        printf("%d\n", nums[i]);
    }
    return 0;
}
```

Exercise 4



```c
#include <stdio.h>

struct Student {
    char name[20];
    int age;
};

int main() {
    struct Student s1 = {"Visal", 18};
    printf("Name: %s\n", s1.name);
    printf("Age: %d\n", s1.age);
    return 0;
}
```

Exercise 5

```c
#include <stdio.h>

int main() {
    int isLoggedIn = 1; // true
    int isAdmin = 0;    // false

    if (isLoggedIn && isAdmin) {
        printf("Access granted\n");
    } else {
        printf("Access denied\n");
    }

    if (!isAdmin) {
        printf("Not an admin\n");
    }

    return 0;
}
```

WHAT HAVE WE LEARNED

- What is JavaScript and how it's used
- console.log() for printing
- Variables: let, const, var
- Data types: number, string, boolean, null, undefined, object
- Operators: arithmetic, comparison, logical
- if, else if, else, ternary operator
- Loops: for, while, do...while, for...in, for...of, forEach()
- Arrays and how to access values

WHAT HAVE WE LEARNED

- Objects and how to access key-value pairs
- Converting C code to JavaScript (syntax & logic differences)

NEXT UP

- Function types:
  - Return + Parameter
  - Return, No Parameter
  - No Return + Parameter
  - No Return, No Parameter
- What is ES6 and why it matters
- Arrow functions
- Default parameters

THANK YOU

REFERENCES

Mozilla Developer Network (MDN). (n.d.). JavaScript guide. Mozilla. Objects and how to access key-value pairs
Converting C code to JavaScript (syntax & logic differences)
W3Schools. (n.d.). JavaScript tutorial. Objects and how to access key-value pairs
Converting C code to JavaScript (syntax & logic differences)
JavaScript.info. (n.d.). The modern JavaScript tutorial. Objects and how to access key-value pairs
Converting C code to JavaScript (syntax & logic differences)
Flanagan, D. (2020). JavaScript: The definitive guide (7th ed.). O'Reilly Media.
Duckett, J. (2014). JavaScript and jQuery: Interactive front-end web development. Wiley.
ECMAScript Language Specification. (n.d.). ECMA-262 Edition 13.0. Ecma International.
Objects and how to access key-value pairs
Converting C code to JavaScript (syntax & logic differences)

KAHOOT TIME

(will provide qr code, once ready)