3 - JavaScript (TypeScript)

React education, 2024.

MITHRIL

Overview

- Basics
- Functions
- Objects
- ES6+
- TypeScript
- Hands-on

MITHRIL

Basics

About JavaScript

- JavaScript is a lightweight interpreted programming language with first-class functions.
- Prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (functional programming) styles.

ECMAScript

- ECMA - European Computer Manufacturers Association
- Scripting language that forms the basis of JavaScript.
- Standardized version of JavaScript - behaves the same way in all applications that support the standard.
    - Set of requirements for implementing standards-compliant language features in your ECMAScript implementation or engine (V8, SpiderMonkey)
    - JavaScript supports all functionality outlined in the ECMAScript specification
- ECMA 15th edition, June 2024

MITHRIL

JS engine

- JS engine is a program, found in a hosting environment such as browser, that executes JS code. It consists of two main parts:
    - **Memory Heap** - a large and unstructured area of memory, objects (function, arrays) are allocated here. No defined limit (except for the hardware limitations of the computer), relatively slow access.
    - **Call Stack** - an abstract data structure located in a special region of the computer's memory, and managed by the CPU. The stack uses LIFO (Last In First Out) principle to temporarily store and manage function invocation calls. Stack grows and shrinks as functions push on and pop off it. It has defined limit (maximum call stack in hosting environment), fast access.
- Popular JS engines: V8 (Chrome, Node.js), SpiderMonkey (Firefox), JavaScriptCore (Safari), Chakra (IE, Edge)

MITHRIL

Grammar and types

- JS is a case-sensitive and uses Unicode character set.
- Comments:
  - // one line comment
  - /* multiline comment */
- Variable declarations:
  - Identifier of a variable must start with a letter, underscore or a dollar sign
  - Local and global variables depending on the execution context - *var*
  - Block-scoped variables - *let*, *const*

MITHRIL

Grammar and types

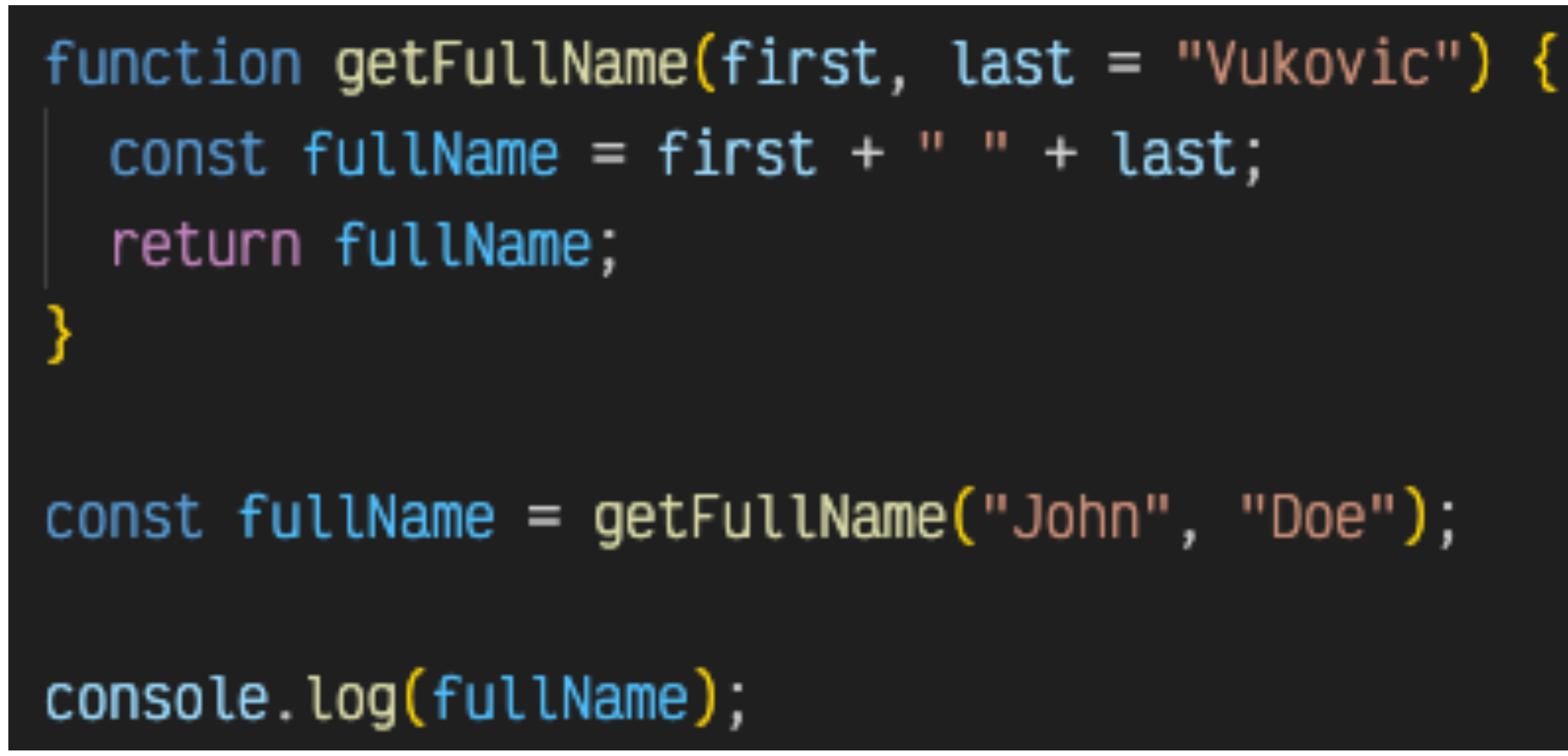
- Latest ECMAScript standard defines eight data types:
  - Boolean - true or false
  - Null - special keyword denoting a null value
  - Undefined - value is not defined
  - Number - an integer or a floating point
  - BigInt - an integer with arbitrary precision
  - String - a sequence of characters
  - Symbol - a data types whose instances are unique and immutable
  - Object - only type that is not primitive
- Read more here.

MITHRIL

Functions

Functions

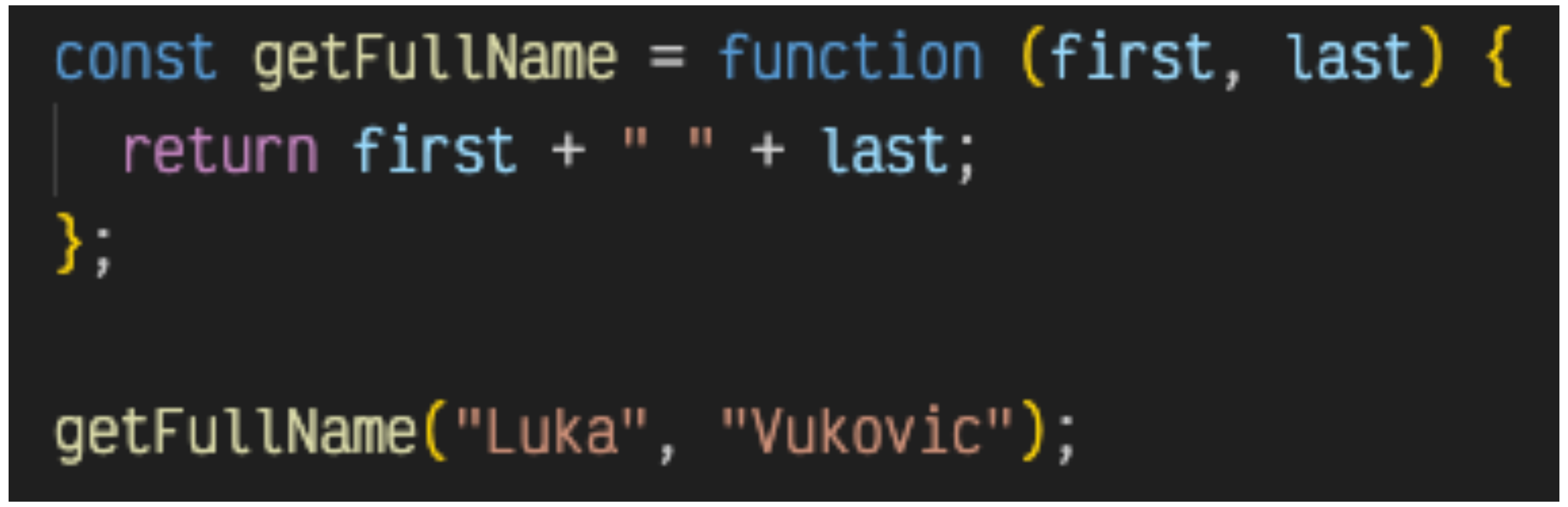
- A function is a JavaScript procedure - a set of statements that performs a task or calculates a value.
- A function definition consists of the function keyword, followed by:
  - The name of the function
  - A list of parameters to the function
  - The JS statements that define the function



```
function getFullName(first, last = "Vukovic") {
  const fullName = first + " " + last;
  return fullName;
}

const fullName = getFullName("John", "Doe");

console.log(fullName);
```

Functions

- Functions can also be created like a function expression.
- Function expression can be named function or an anonymous which means it doesn't have to have a name.

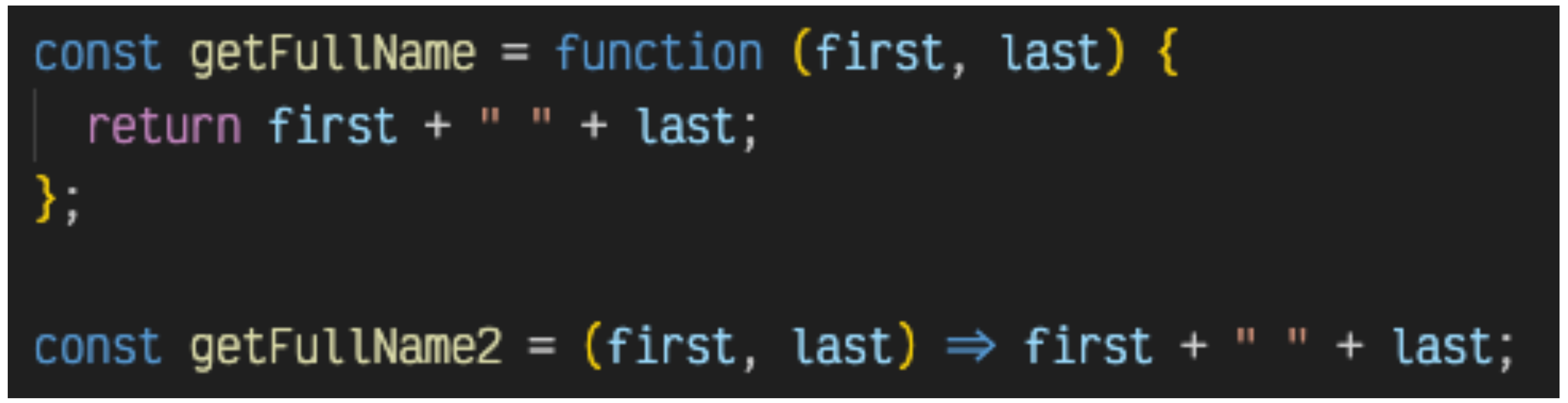


```
const getFullName = function (first, last) {
  return first + " " + last;
};

getFullName("Luka", "Vukovic");
```

MITHRIL

Arrow functions

- An arrow function has a shorter syntax compared to function expressions and does not have its own this, super.
- Two factors influenced the introduction of arrow functions:
  - Shorter syntax
  - Non-binding of **this**
- Read more here.



```
const getFullName = function (first, last) {
  return first + " " + last;
};

const getFullName2 = (first, last) => first + " " + last;
```

Objects

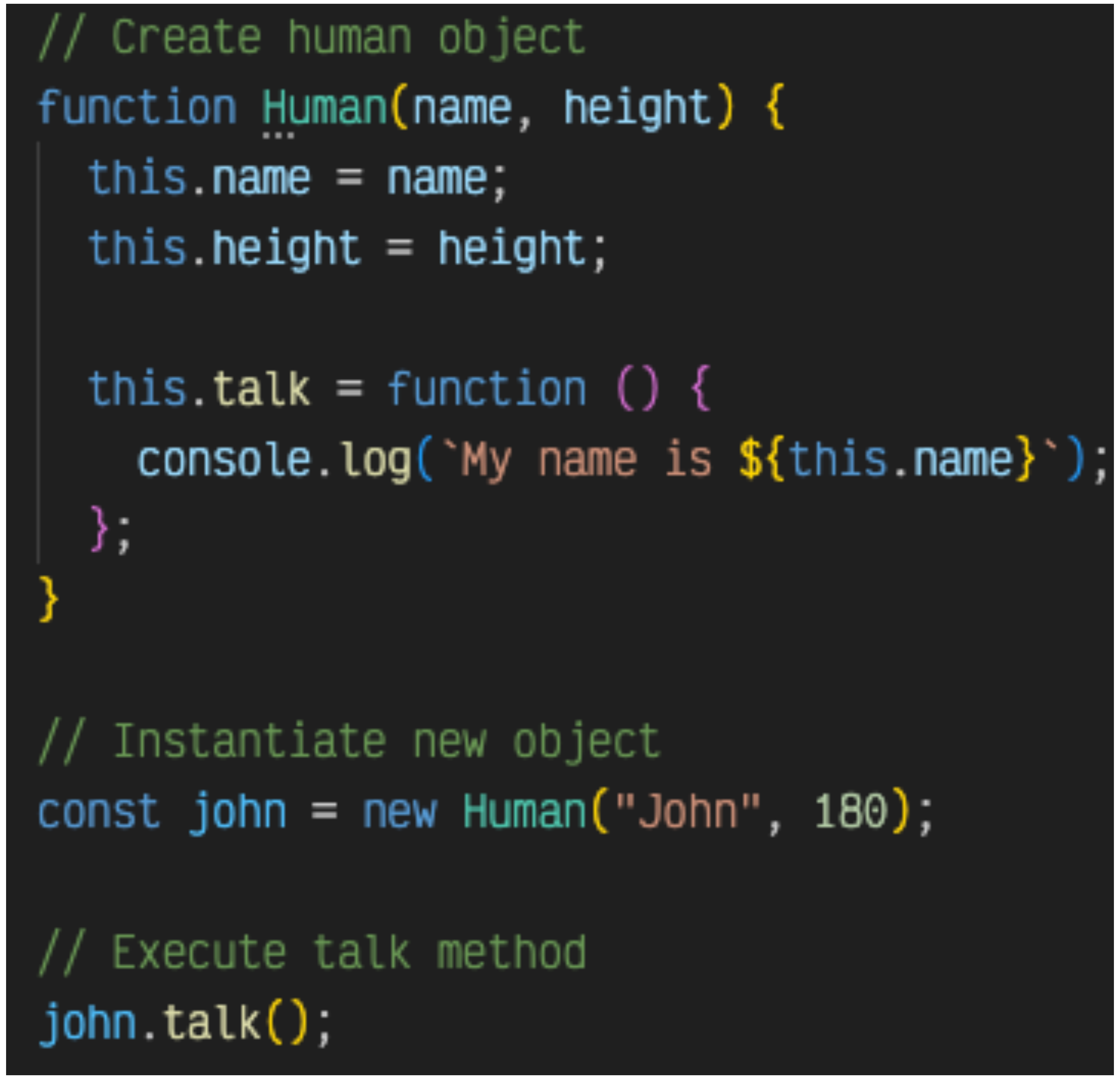
Objects

- JavaScript is designed on a simple object-based paradigm. An object is a collection of properties and property is an association between a name (or key) and value.
- A property’s value can be a function, in which case the property is known as a method.
- An object is a standalone entity, with properties that define entity’s characteristics.

MITHRIL

Objects

- Code on the right side shows example of JS object.
- Read more here.



```
// Create human object
function Human(name, height) {
  this.name = name;
  this.height = height;

  this.talk = function () {
    console.log(`My name is ${this.name}`);
  };
}

// Instantiate new object
const john = new Human("John", 180);

// Execute talk method
john.talk();
```

ES6+

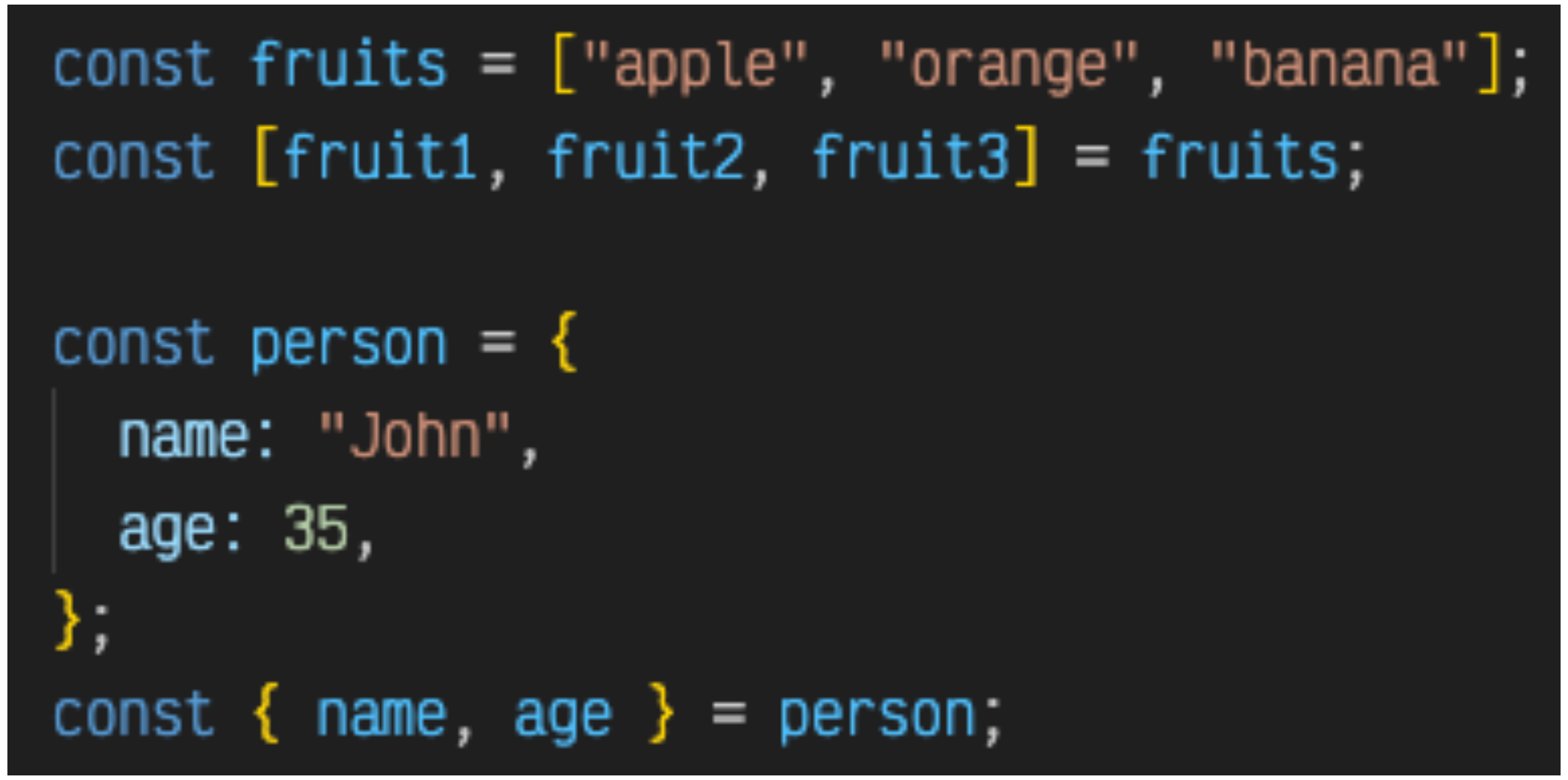
ES6+

- Some of the new JS features:
  - Destructuring
  - Spread operator
  - Template literals
  - Async/await

MITHRIL

ES6+

- **Destructuring** allows us to extract data from arrays or objects into distinct variables in an easy way.

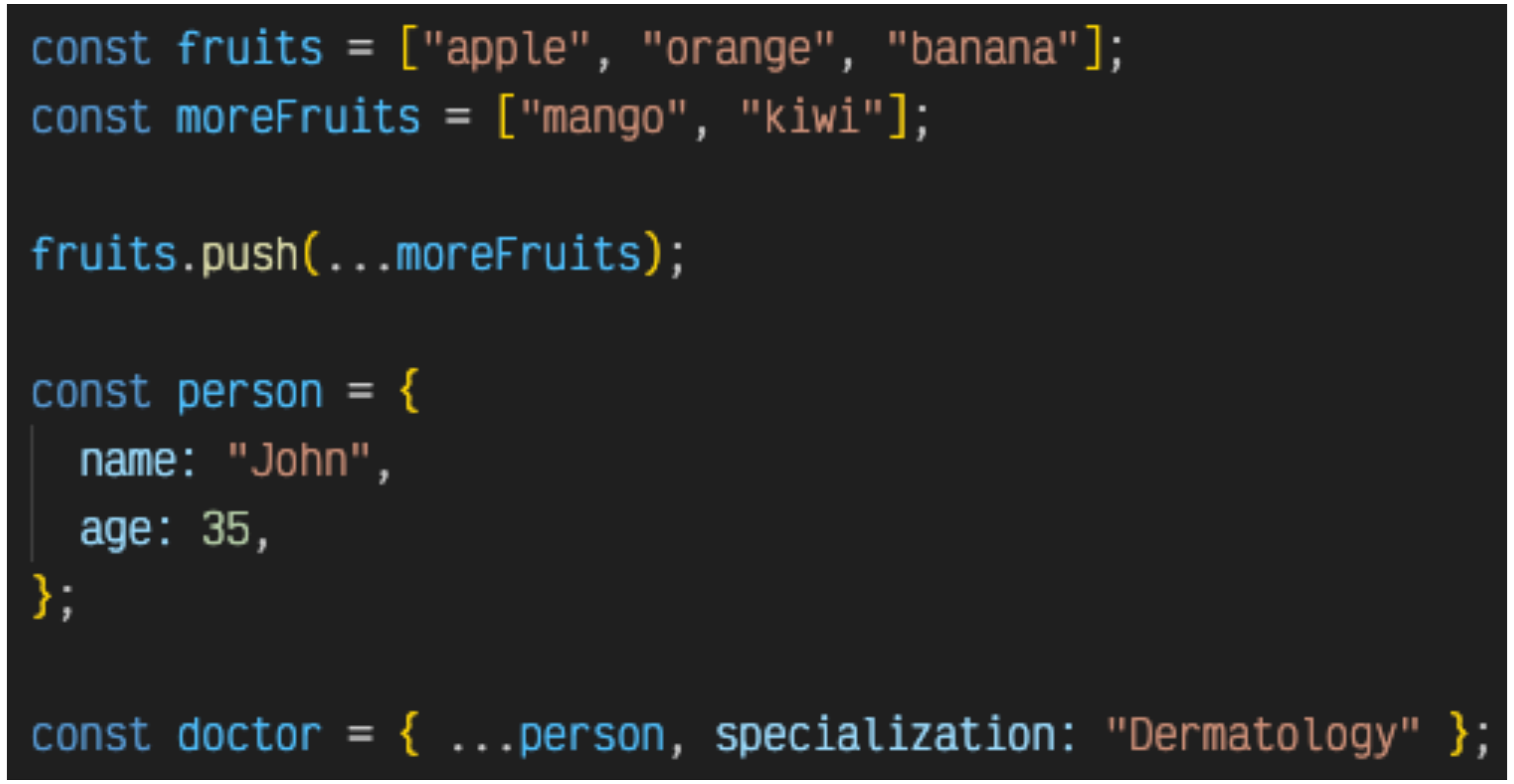


```
const fruits = ["apple", "orange", "banana"];
const [fruit1, fruit2, fruit3] = fruits;

const person = {
  name: "John",
  age: 35,
};
const { name, age } = person;
```

MITHRIL

ES6+

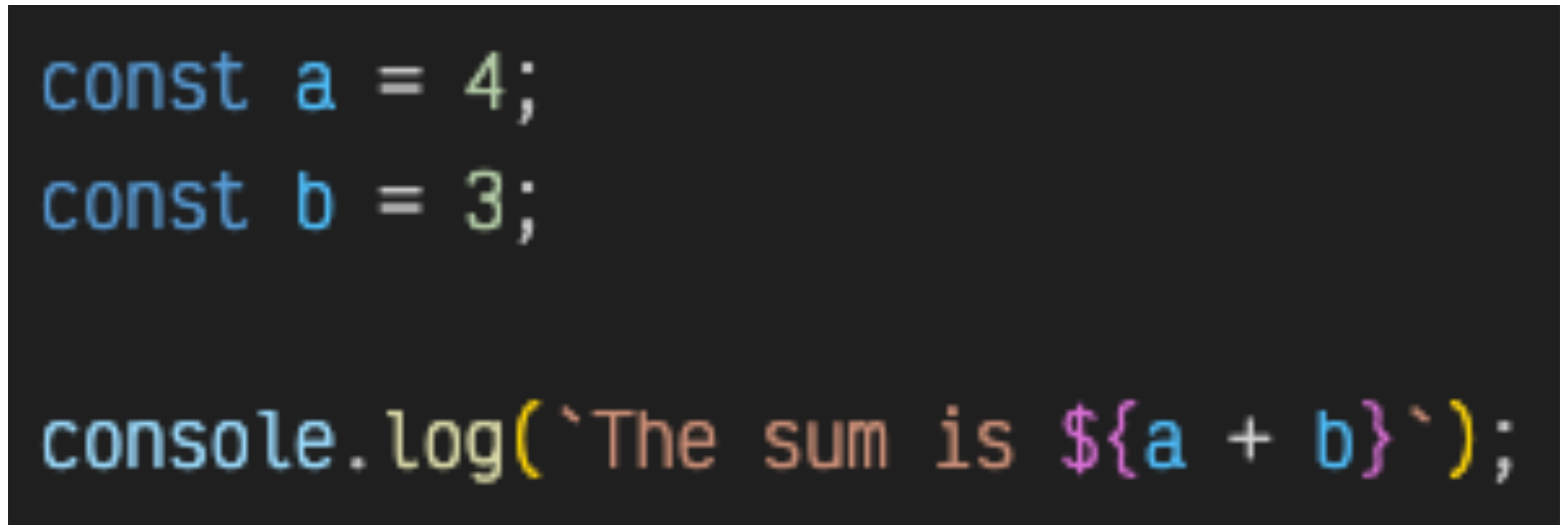
- **Speed operator** allows us to explode an array into its individual items.
- With spread operator we can copy values from the object.



```
const fruits = ["apple", "orange", "banana"];
const moreFruits = ["mango", "kiwi"];

fruits.push(...moreFruits);

const person = {
  name: "John",
  age: 35,
};

const doctor = { ...person, specialization: "Dermatology" };
```

ES6+

- **Template literals** are enclosed by backtick ` ` character which allows us to contain placeholders ${} and write strings in multiple rows.

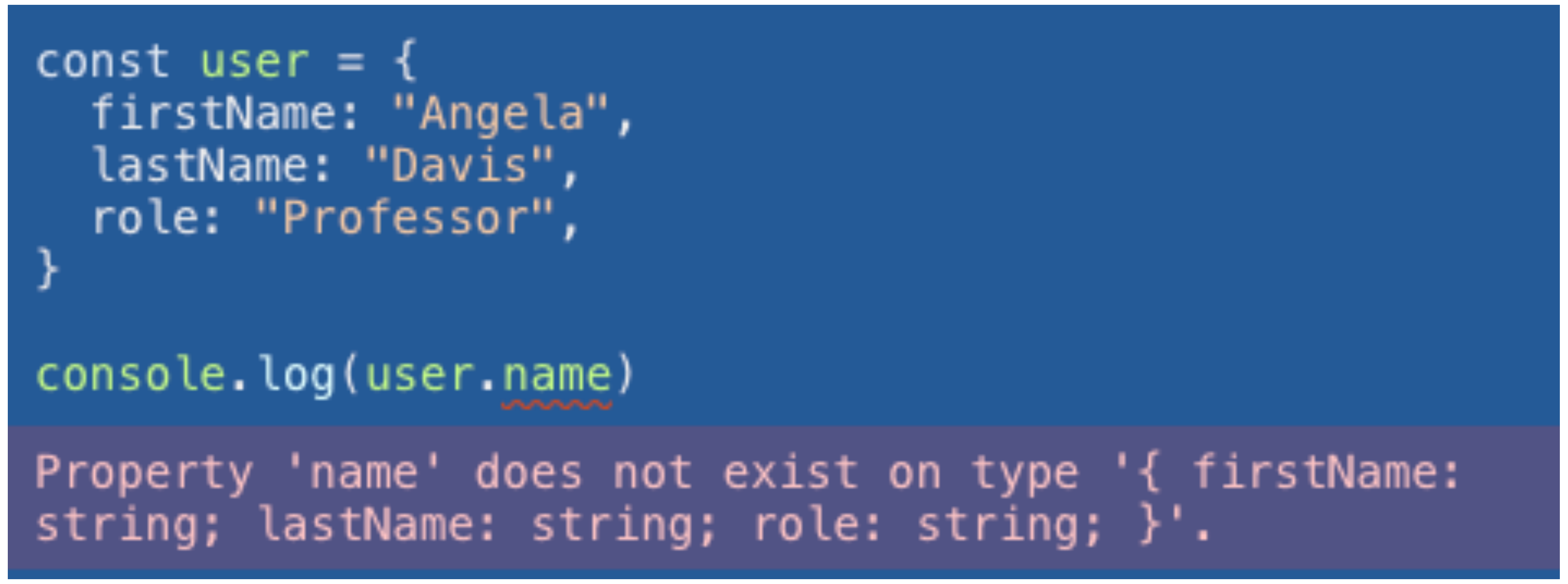


```
const a = 4;
const b = 3;

console.log(`The sum is ${a + b}`);
```

TypeScript

TypeScript

- TypeScript is JavaScript with syntax for types.
- TypeScript code converts to JavaScript, which runs anywhere JavaScript runs: in a browser, on Node.js or Deno and in your apps.
- TypeScript add additional syntax to JavaScript to support a tighter integration with your editor. You are able to catch error early in your editor.



```
const user = {
  firstName: "Angela",
  lastName: "Davis",
  role: "Professor",
}

console.log(user.name)
```

```
Property 'name' does not exist on type '{ firstName:
string; lastName: string; role: string; }'.
```

Hands-on

Hands-on

- Create a JavaScript console application locally or inside playcode.
- Create new object - create a driver object with properties: name (string), time (integer), topSpeed (integer). Objects need to be created in a way so you can instantiate any number of other objects from it.
- Instantiate three new objects - each new object needs to have all three properties defined.
- Create array of objects - put newly created objects inside the array.
- Create function *getRaceStatistics* that accepts array of driver objects as an argument. It prints drivers objects sorted from fastest to slowest. Read more about sort method here.
- Try to print drivers in following format:

1. (name) finished race in (time) seconds with top speed of (topSpeed) km/h

2. (name) finished race in (time) seconds with top speed of (topSpeed) km/h

3. (name) finished race in (time) seconds with top speed of (topSpeed) km/h

MITHRIL