BAITUSSALAM

BAITUSSALAM TECH PARK

Class Agenda

BAITUSSALAM
—TECH PARK—

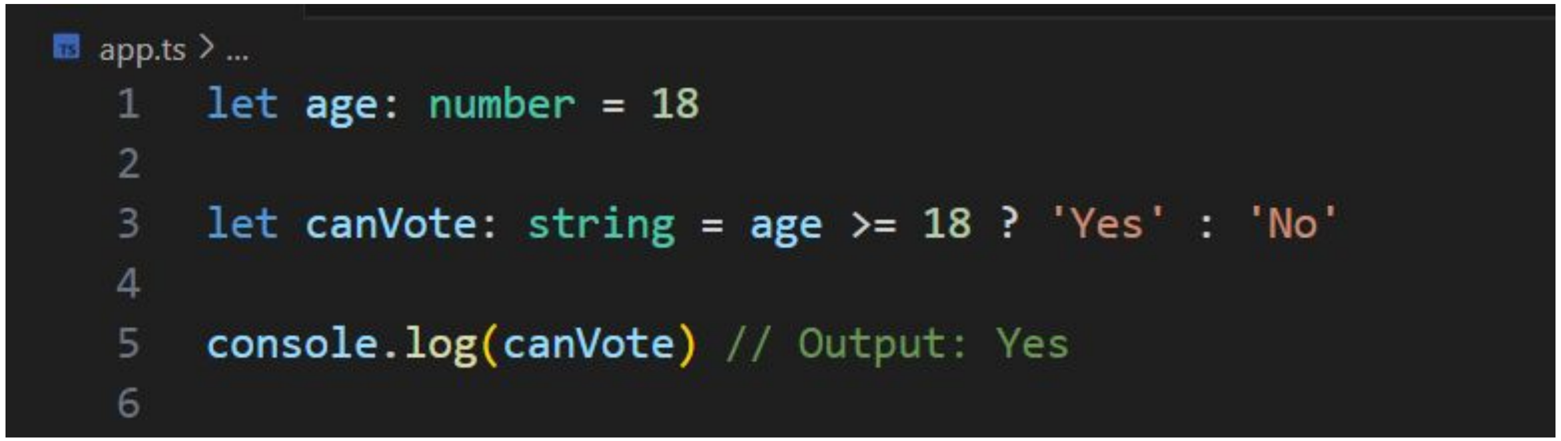
Ternary, Rest and Spread Operator, reduce , Local Storage and Session Storage

Ternary Operator

What is Ternary Operator?

- The ternary operator is a shorthand for the if-else statement.
- It is also known as the conditional operator.

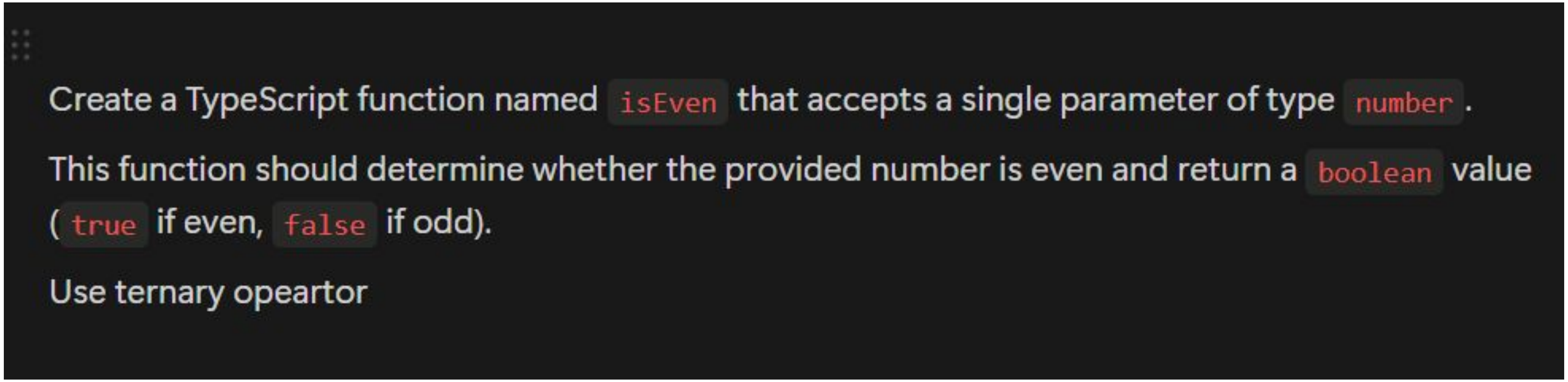
Syntax: condition ? expressionIfTrue : expressionIfFalse



app.ts > ...

```
let age: number = 18

let canVote: string = age >= 18 ? 'Yes' : 'No'

console.log(canVote) // Output: Yes

```

Ternary Operator Exercise



Create a TypeScript function named `isEven` that accepts a single parameter of type `number`.

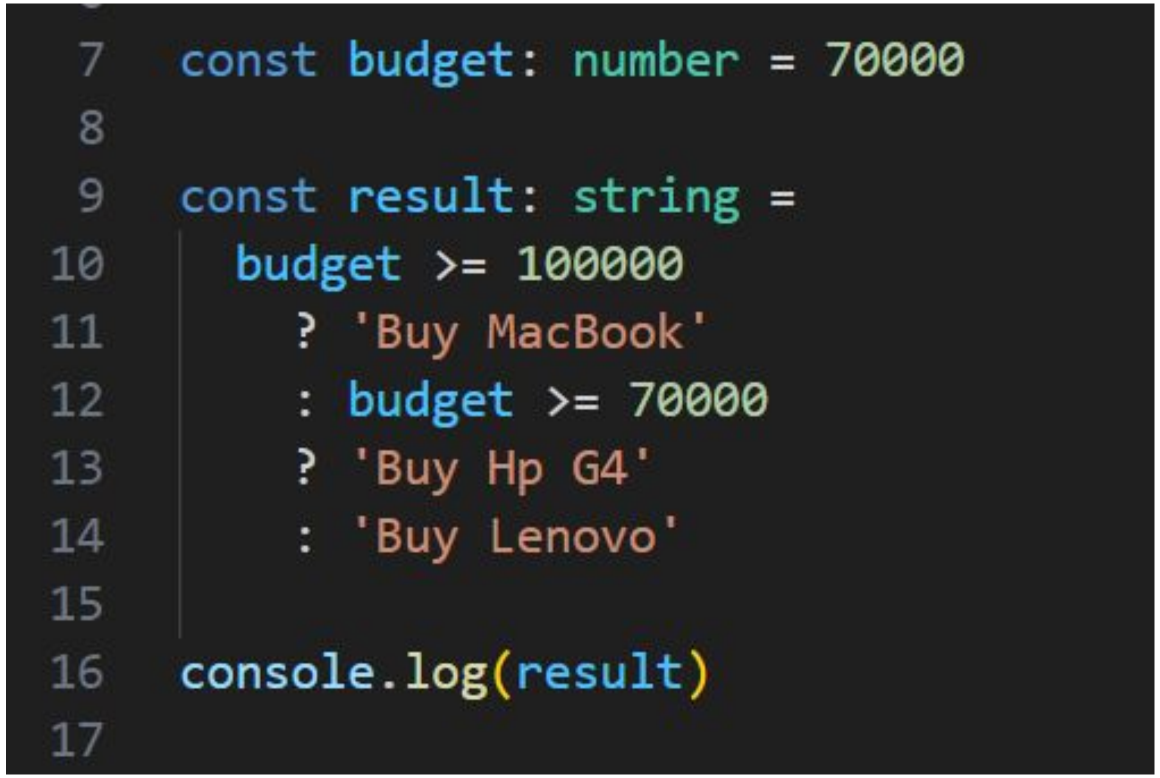
This function should determine whether the provided number is even and return a `boolean` value (`true` if even, `false` if odd).

Use ternary opeartor

Nested Ternary Operator

What is Nested Ternary Operator?

- A ternary operator inside another ternary operator.
- Used for multiple conditions in a concise way.

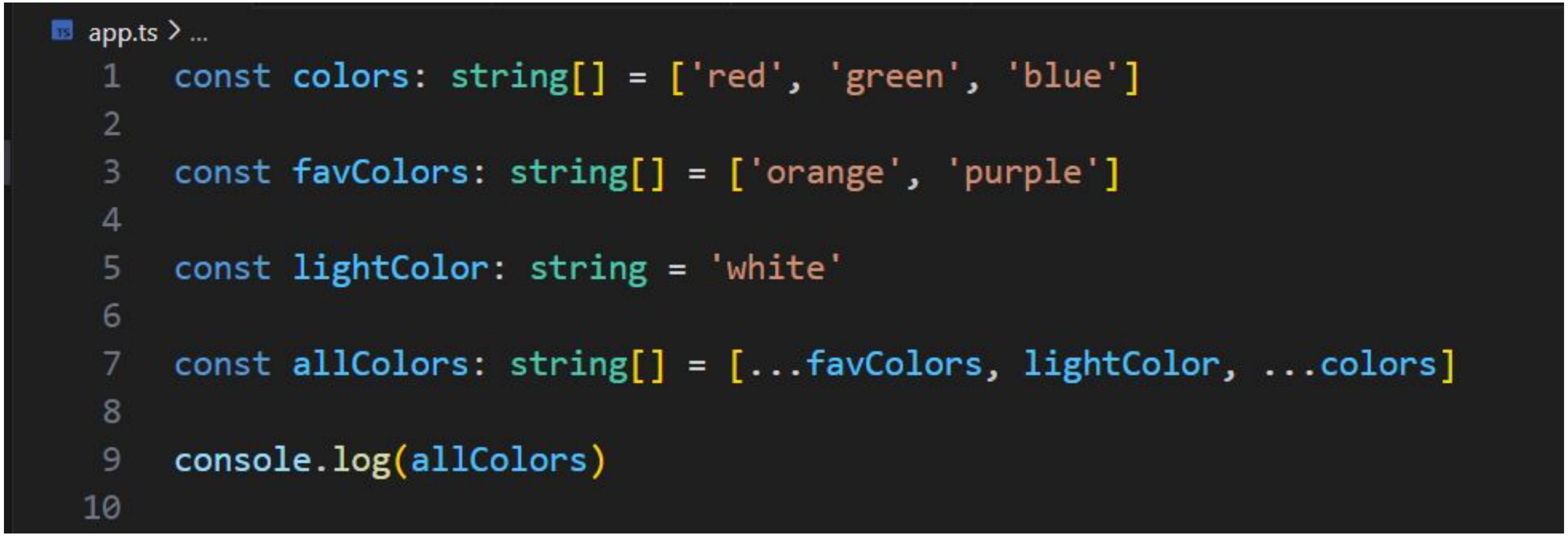


```
const budget: number = 70000

const result: string =
  budget >= 100000
    ? 'Buy MacBook'
    : budget >= 70000
    ? 'Buy Hp G4'
    : 'Buy Lenovo'

console.log(result)

```

Rest and Spread Operator

Spread operator: The spread (...) syntax lets you take all the items in an array or string and use them in places where you need individual items, like when calling a function or creating a new array. For objects, the spread syntax takes all the properties from one object and adds them to another object you're creating.

Spread Operator with Array

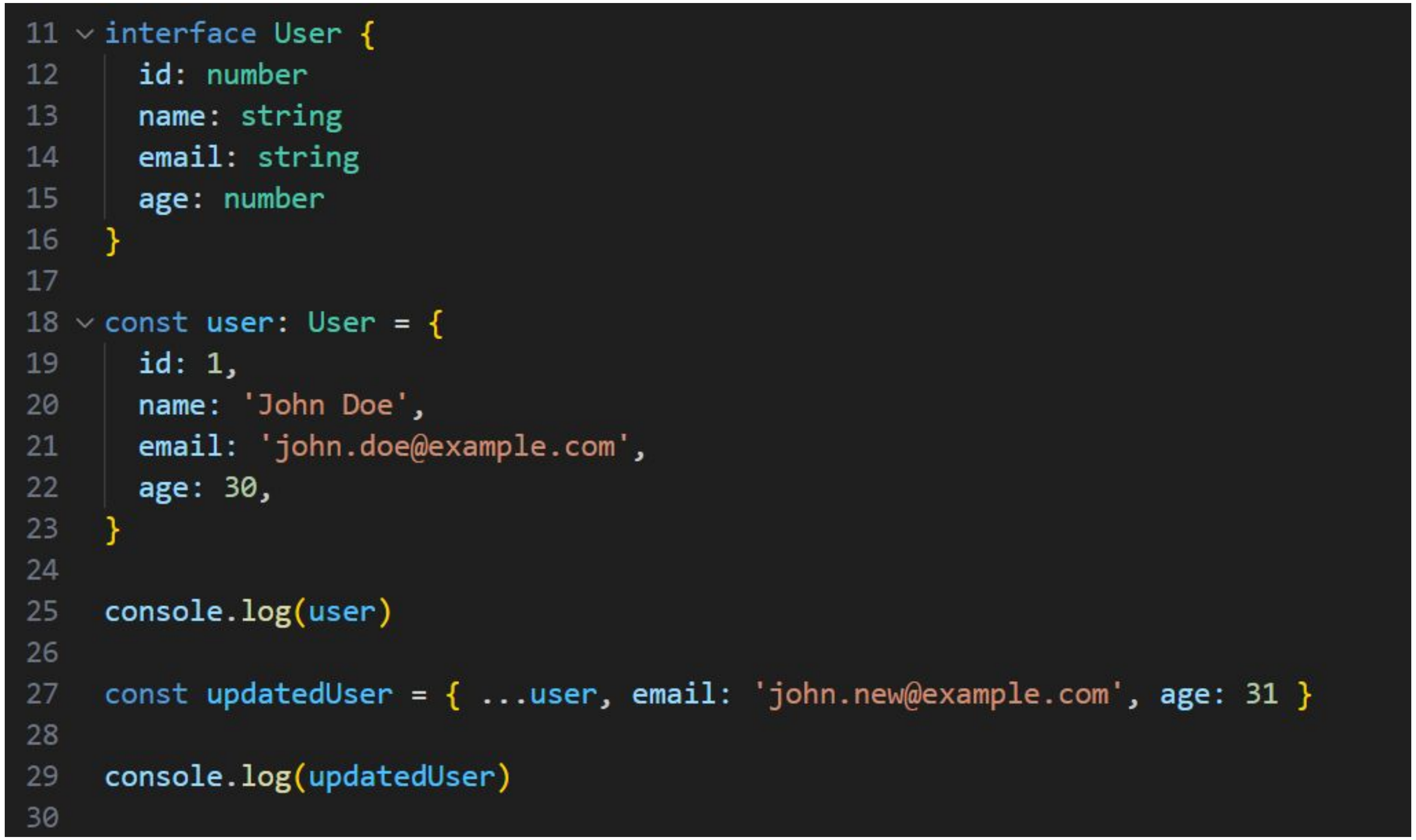


app.ts > ...

```
const colors: string[] = ['red', 'green', 'blue']

const favColors: string[] = ['orange', 'purple']

const lightColor: string = 'white'

const allColors: string[] = [...favColors, lightColor, ...colors]

console.log(allColors)

```

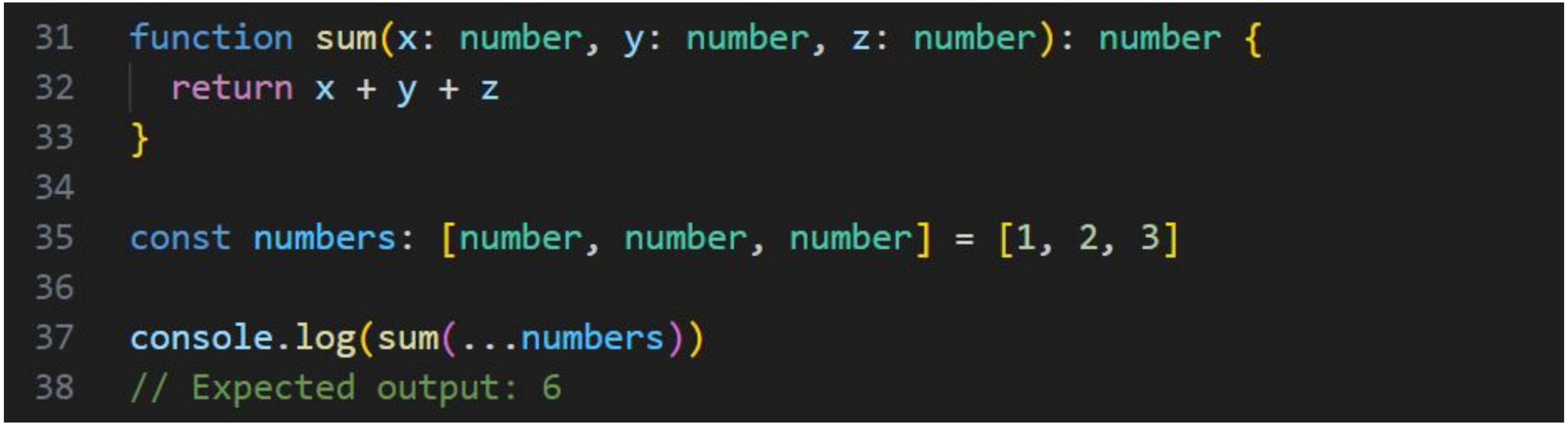
Spread Operator with Object

Cloning and updating object



```
interface User {
  id: number
  name: string
  email: string
  age: number
}

const user: User = {
  id: 1,
  name: 'John Doe',
  email: 'john.doe@example.com',
  age: 30,
}

console.log(user)

const updatedUser = { ...user, email: 'john.new@example.com', age: 31 }

console.log(updatedUser)
```

Spread Operator with Function



```
function sum(x: number, y: number, z: number): number {
  return x + y + z
}

const numbers: [number, number, number] = [1, 2, 3]

console.log(sum(...numbers))
// Expected output: 6
```

Spread Operator Exercise #1

Create a typescript function **mergeArrays** that merge two arrays using spread operator

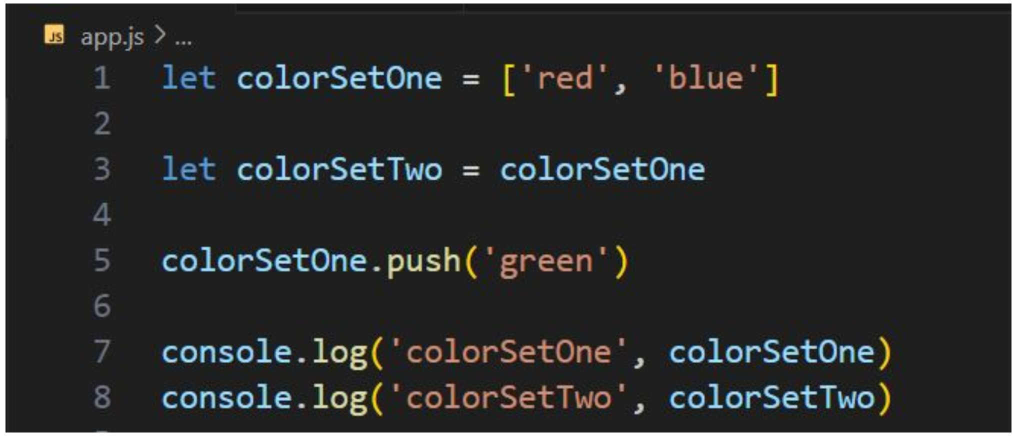
Example

```
const array1: number[] = [1, 2, 3]
const array2: number[] = [4, 5, 6]
```

Expected output

```
mergeArrays(array1, array2)  // [1, 2, 3, 4, 5, 6]
```

Most important Array Concept



app.js > ...

```js
let colorSetOne = ['red', 'blue']

let colorSetTwo = colorSetOne

colorSetOne.push('green')

console.log('colorSetOne', colorSetOne)
console.log('colorSetTwo', colorSetTwo)
```

In JavaScript, when we assign an array (or any object) to another variable, we're not creating a new copy of that array; instead, we're creating a **reference to the same array** in memory.

So, when we modify the array through one reference, the change is reflected in all other references pointing to that array.

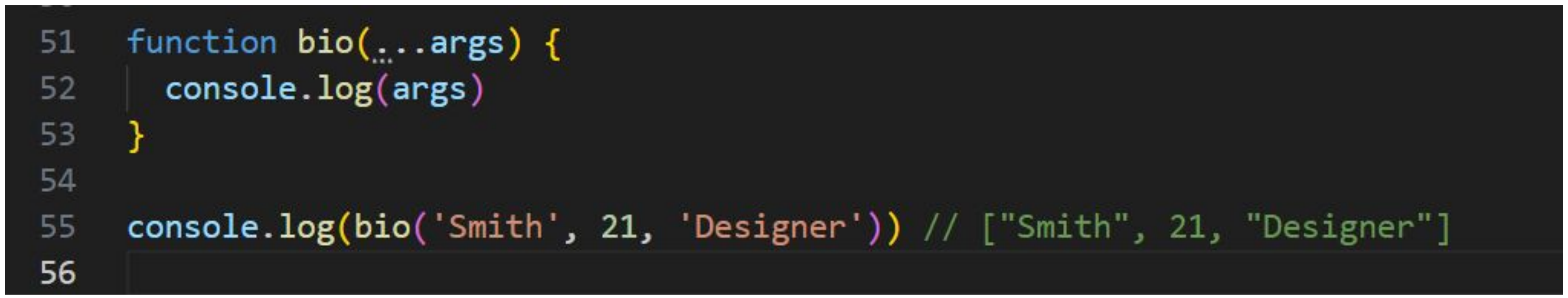
A **shallow copy** means that it creates a new array, but the elements within that array are still references to the same objects as the original array.

Rest Parameters

In JavaScript functions, rest gets used as a prefix of the function's last parameter.

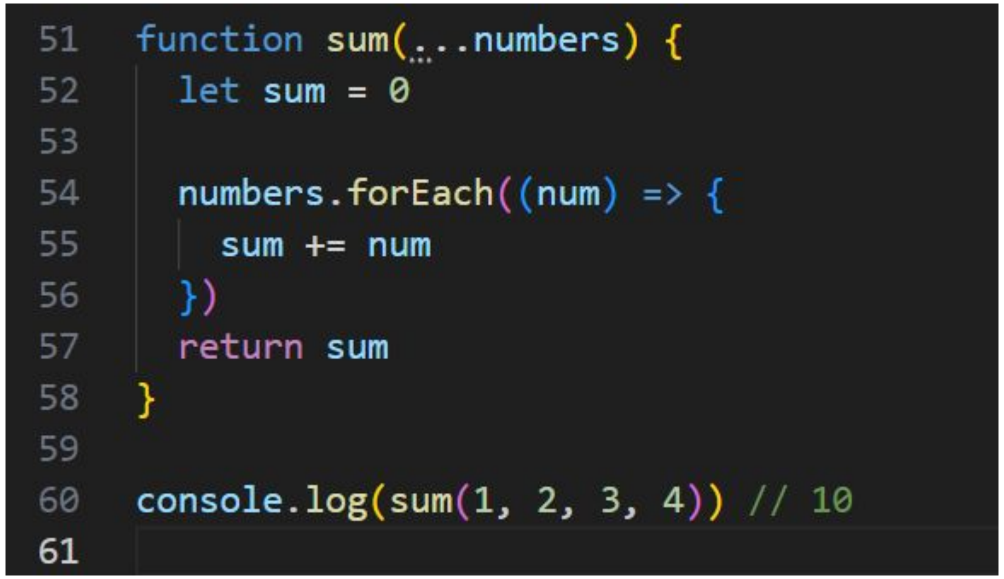
- Rest parameter (...args) allows functions to accept an indefinite number of arguments as an array.
- Useful for functions where the number of parameters may vary.

Rest Parameters Example



```
function bio(...args) {
  console.log(args)
}

console.log(bio('Smith', 21, 'Designer')) // ["Smith", 21, "Designer"]
```

Syntax

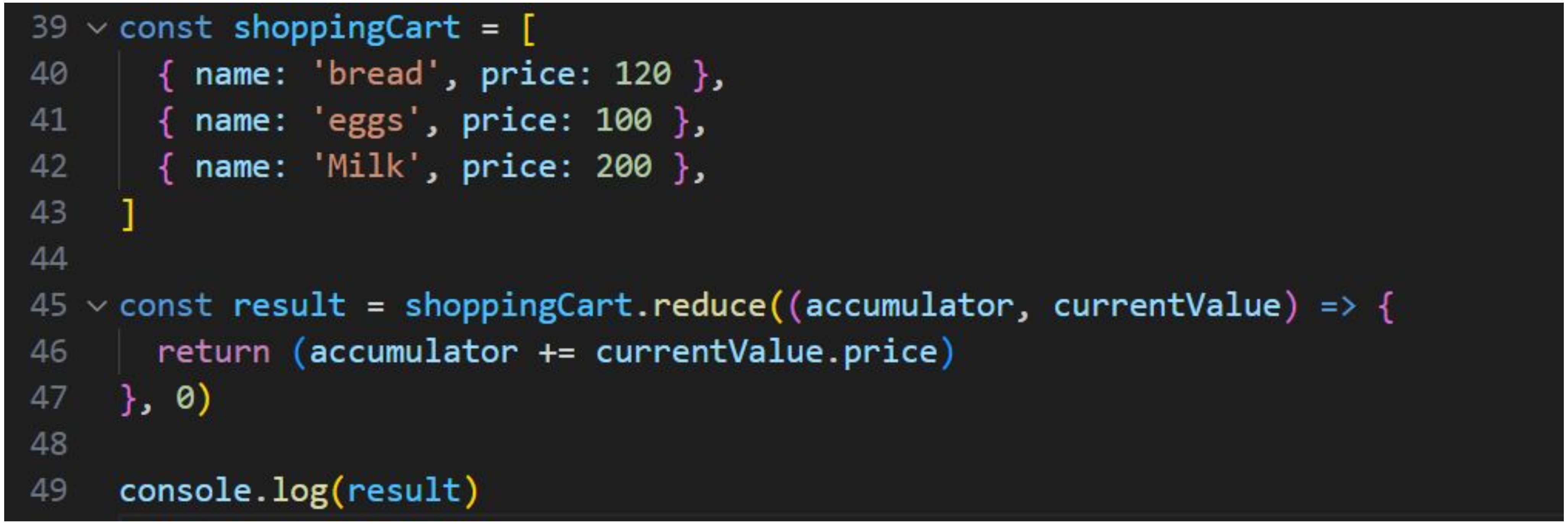


```
function sum(...numbers) {
  let sum = 0

  numbers.forEach((num) => {
    sum += num
  })
  return sum
}

console.log(sum(1, 2, 3, 4)) // 10
```

Example

Reduce Method

The reduce() method got its name from the functionality it provides, which is to iterate and “reduce” an array's values into one value.



```javascript
const shoppingCart = [
  { name: 'bread', price: 120 },
  { name: 'eggs', price: 100 },
  { name: 'Milk', price: 200 },
]

const result = shoppingCart.reduce((accumulator, currentValue) => {
  return (accumulator += currentValue.price)
}, 0)

console.log(result)
```

Web Storage

What is Web Storage?

Web Storage provides a way to store data in the browser.

Types:

- Local Storage
- Session Storage

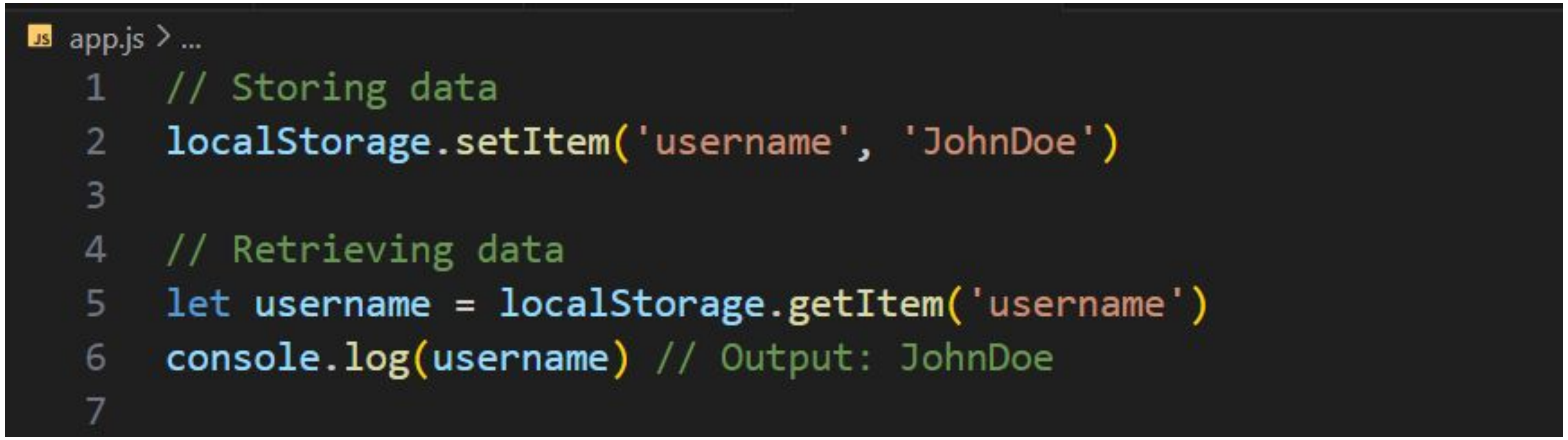
Key Features:

- Stores data as key-value pairs.
- Data is stored as strings.

Local Storage

What is Local Storage?

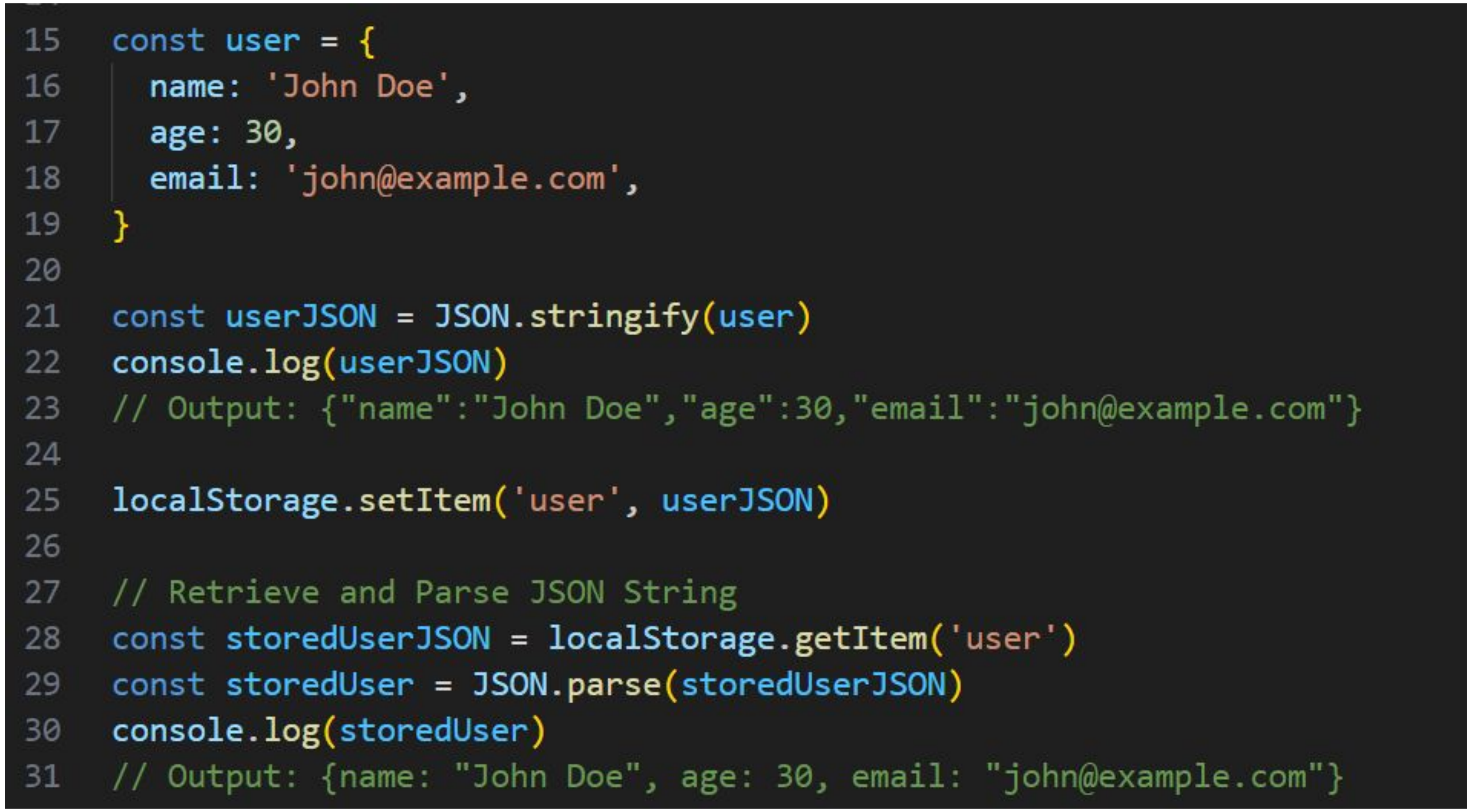
- Local Storage allows you to store data with no expiration date.
- Data persists even when the browser is closed and reopened.


app.js > ...

```javascript
// Storing data
localStorage.setItem('username', 'JohnDoe')

// Retrieving data
let username = localStorage.getItem('username')
console.log(username) // Output: JohnDoe

```

Storing Object in Local Storage

- Convert object to JSON string using JSON.stringify().
- Store JSON string in local storage using localStorage.setItem().
- Retrieve JSON string from local storage using localStorage.getItem().
- Convert JSON string back to an object using JSON.parse().

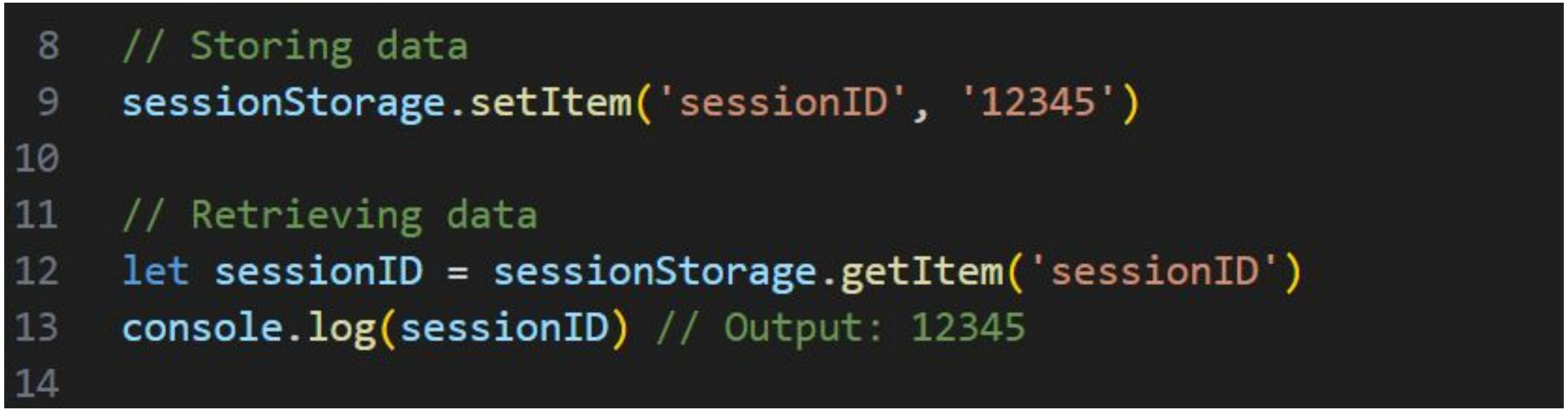


```
const user = {
  name: 'John Doe',
  age: 30,
  email: 'john@example.com',
}

const userJSON = JSON.stringify(user)
console.log(userJSON)
// Output: {"name":"John Doe","age":30,"email":"john@example.com"}

localStorage.setItem('user', userJSON)

// Retrieve and Parse JSON String
const storedUserJSON = localStorage.getItem('user')
const storedUser = JSON.parse(storedUserJSON)
console.log(storedUser)
// Output: {name: "John Doe", age: 30, email: "john@example.com"}
```

Session Storage

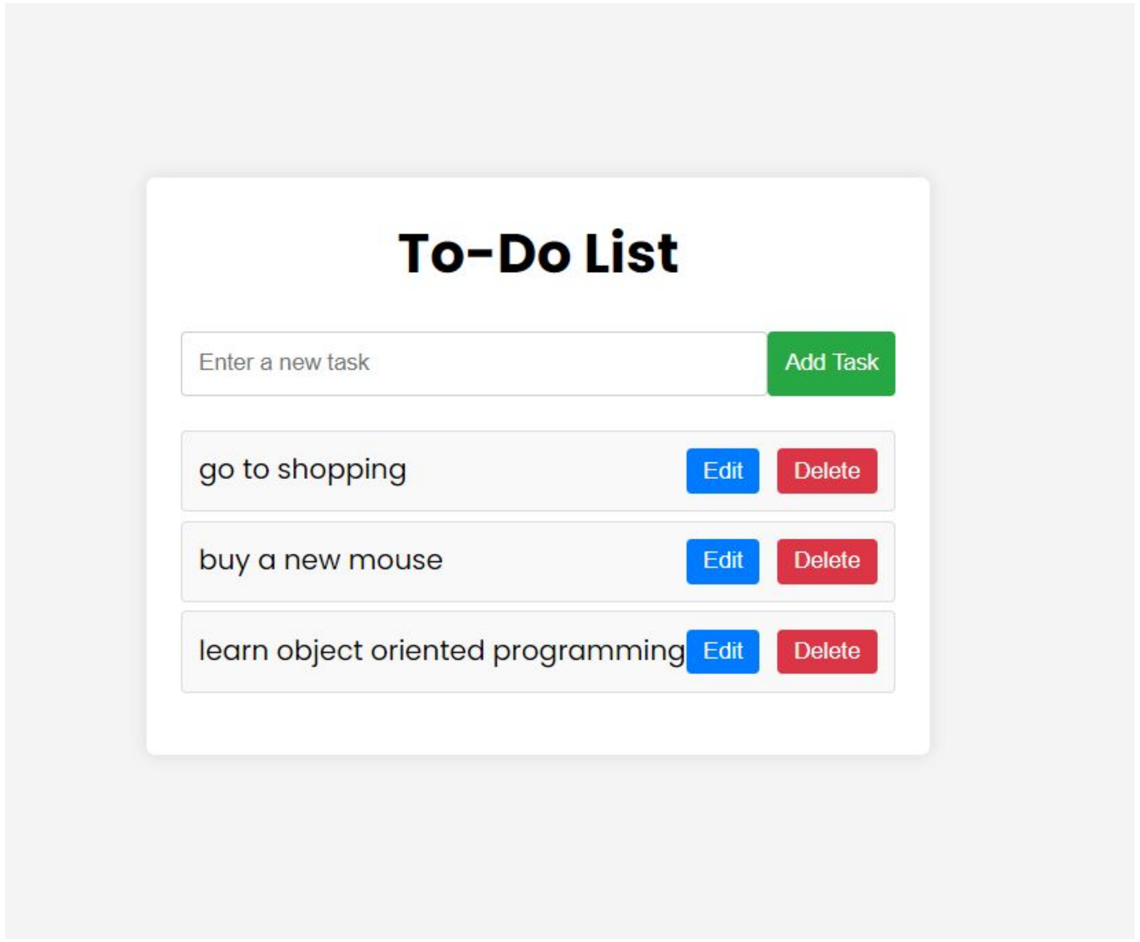
What is Session Storage?

- Session Storage stores data for the duration of the page session.
- Data is cleared when the page session ends (e.g., tab or window is closed).



```javascript
// Storing data
sessionStorage.setItem('sessionID', '12345')

// Retrieving data
let sessionID = sessionStorage.getItem('sessionID')
console.log(sessionID) // Output: 12345

```

Task List App with Local Storage



To-Do List

Enter a new task

Add Task

go to shopping

Edit Delete

buy a new mouse

Edit Delete

learn object oriented programming

Edit Delete

The End

BAITUSSALAM
—TECH PARK—