Data Types in JavaScript

1. Primitive Data Types

1. Number
2. String
3. Boolean
4. Null
5. Undefined
6. Symbol
7. BigInt

1. Number

The Number data type represents both integer and floating-point numbers. It is used to perform mathematical calculations and store numeric values.

```
let age = 25;
let pi = 3.14;
```

2. String

The String data type represents a sequence of characters. It is used to store and manipulate textual data. Strings are enclosed in single quotes ('') or double quotes ("")

```
let fullName = 'Uzair Ahmad';
let message = "Hello, World!";
```

3. Boolean

The Boolean data type represents a logical value, either true or false. Booleans are commonly used in conditional statements and logical operations.

```
let isLogged = true;
let hasPermission = false;
```

4. Null

The Null data type represents the intentional absence of any object value. It is often used to indicate the absence of a meaningful value.

```
let value = null;
```

5. Undefined

The Undefined data type represents a variable that has been declared but has not been assigned a value. It is the default value of uninitialized variables.

```
let fullName ;
let age ;
```

6. Symbol

The Symbol data type represents a unique and immutable value. Symbols are primarily used as property keys in objects to avoid naming conflicts.

```
let id = Symbol('unique id');
```

7. BigInt

The BigInt data type is used to represent arbitrarily large integers with precision. It is particularly useful when dealing with numbers that exceed the maximum safe integer value in JavaScript (Number.MAX_SAFE_INTEGER, which is 2^53 - 1).

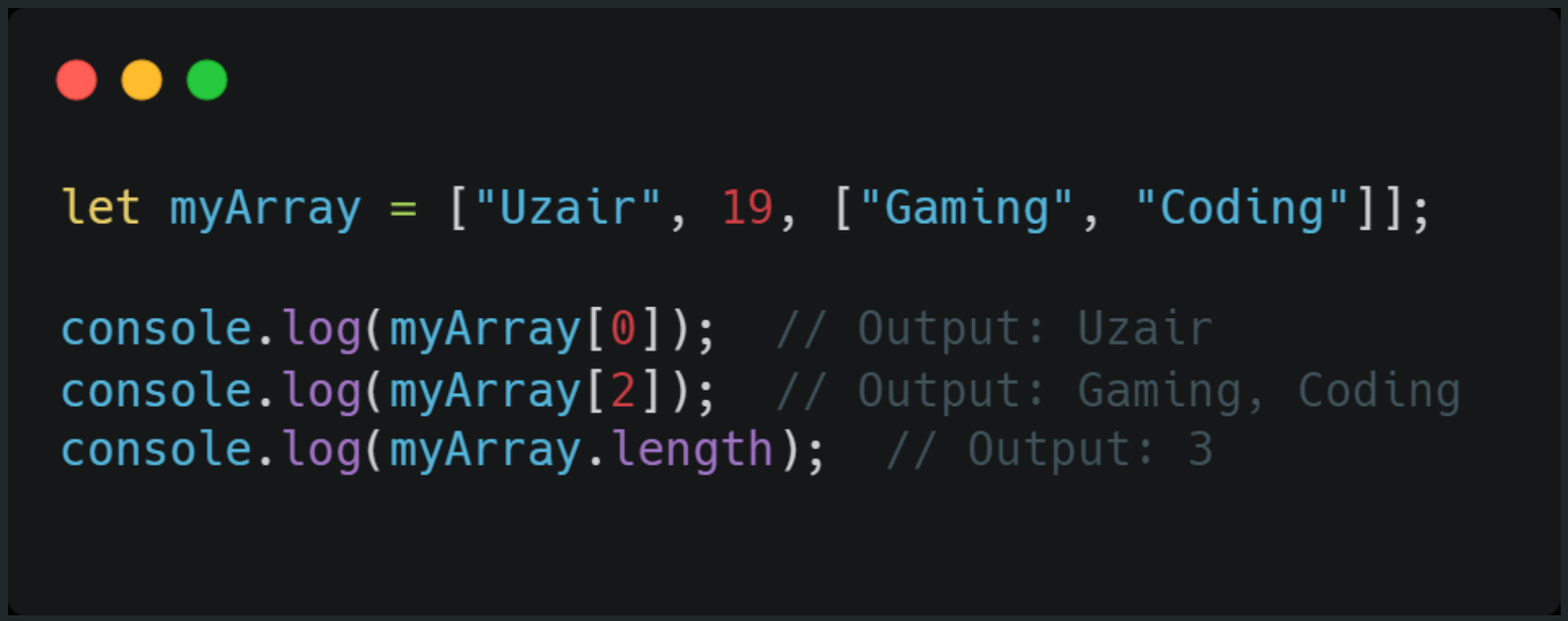
```
let bigNumber = 12345678901234567890123456789017890n;
let bigNumberFromFunction = BigInt("98765432109876547654321O");
```

2. Composite Data Types

1. Array
2. Object
3. Function
4. Set
5. Map

1. Array

An array is an ordered list of values represented by square brackets ([]). It allows for storing multiple values of different types, such as numbers, strings, objects, or even other arrays. Arrays can be accessed and manipulated using various methods and properties. Here's an example:



Arrays are zero-indexed, meaning the first element is accessed with an index of 0, the second element with an index of 1, and so on. To access a specific element in an array, you can use square brackets and provide the index within them.

2. Object

Objects are collections of key-value pairs, represented by curly braces ({}). They allow you to group related data and functions together. Object properties can hold values of any data type, including other objects and arrays. Properties are accessed using dot notation (.) or square brackets ([])

Example of Object

```
let person = {
  name: 'Uzair',
  age: 25,
  hobbies: ['gaming', 'coding'],
  greet: function() {
    console.log('Hello!');
  }
};

console.log(person.name);  // Output: "Uzair"
person.greet();  // Output: "Hello!"
```

3. Function

Functions are reusable blocks of code that perform a specific task. They can take arguments, execute a set of instructions, and optionally return a value. Functions can be declared using the function keyword or as arrow functions (introduced in ES6)

```
function addNumbers(a, b) {
  return a + b;
}
let result = addNumbers(3, 4);  // Output: 7
```

4. Set

Sets are collections of unique values, providing an easy way to manage distinct elements. They can store any type of value and automatically handle uniqueness. Sets offer methods for adding, removing, and checking the presence of elements

Example of Set

```
let mySet = new Set();
mySet.add(1);
mySet.add(2);
mySet.add(2);  // Ignored (already exists)
console.log(mySet.size);  // Output: 2
mySet.delete(2);
console.log(mySet.has(2));  // Output: false
```

5. Map

Maps are collections of key-value pairs, similar to objects. However, maps allow any data type as keys and maintain the order of insertion. They offer methods for adding, removing, and retrieving values based on the keys.

Example of Map

```
let myMap = new Map();

myMap.set('name', 'Uzair');

myMap.set('age', 25);

console.log(myMap.get('name'));  // Output: "Uzair"
console.log(myMap.has('age'));  // Output: true

myMap.delete('age');
console.log(myMap.has('age'));  // Output: false
```

Like

Share

Follow

Uzair Ahmad

Save for Later