Why arrays in JavaScript dynamic ?

Introduction

An **array** is a collection of items stored at contiguous memory locations, meaning items are stored consecutively in the memory.

Each item can be accessed through its **index** (position) number. Arrays always start at index 0.

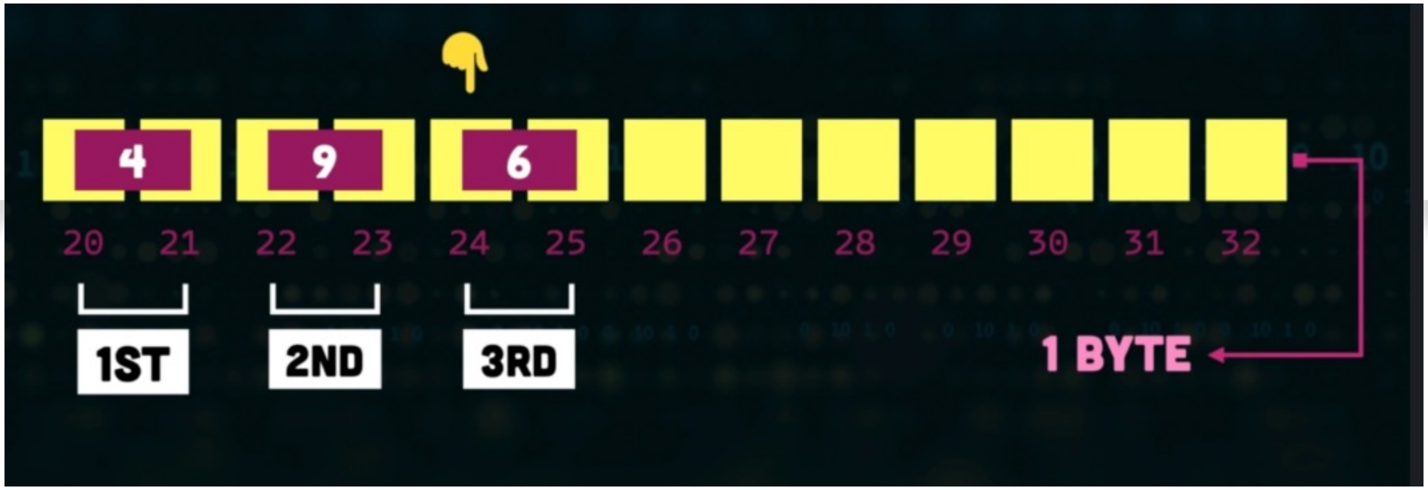
Key differences between arrays in JavaScript and a strongly-typed language like Java or C++:

JavaScript arrays can hold elements of different data types in the same array. For example, you can have numbers, strings, objects, and other data types all in one array.	In Java or C++, arrays are typically homogeneous, meaning all elements must have the same data type. If you declare an array of integers, it can only store integers.
Arrays in JavaScript are dynamic and don't require a fixed size or type declaration when created. You can add or remove elements freely without specifying a size in advance.	In Java or C++, you often need to declare the size and type of an array when defining it. This is typically done at compile-time.

Why arrays in JavaScript can have mixed types?

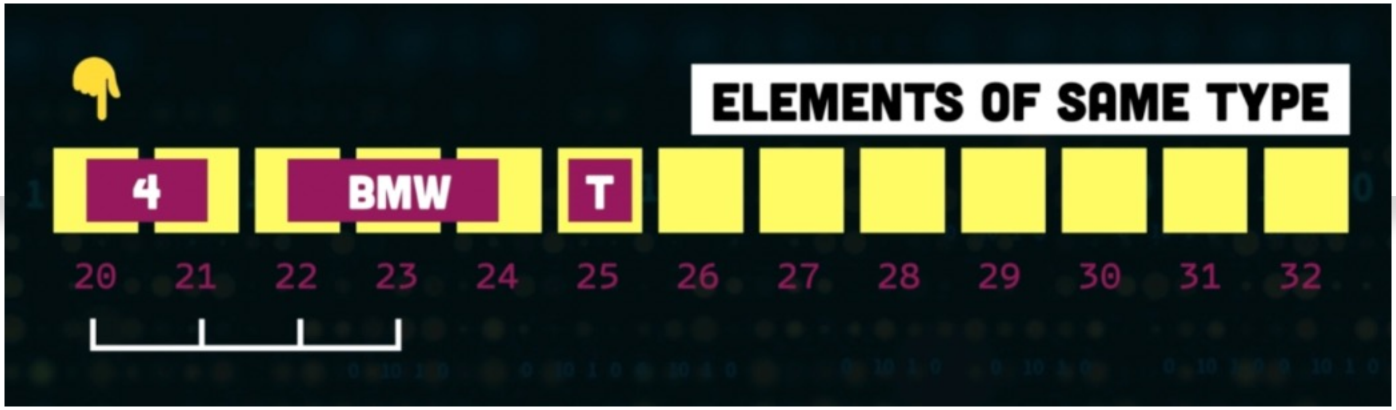
In JavaScript, we can have an array of mixed types with dynamic size. This is because JavaScript is a dynamically typed language, which means that the type of a value is determined at runtime rather than at compile time. This allows JavaScript to have arrays that can store values of different types because the type of an element can be changed at any time.

In a strongly-typed language, homogenous data can only be stored in arrays. Below represents the memory of an array of integers (each integer takes up 2 bytes of memory) :



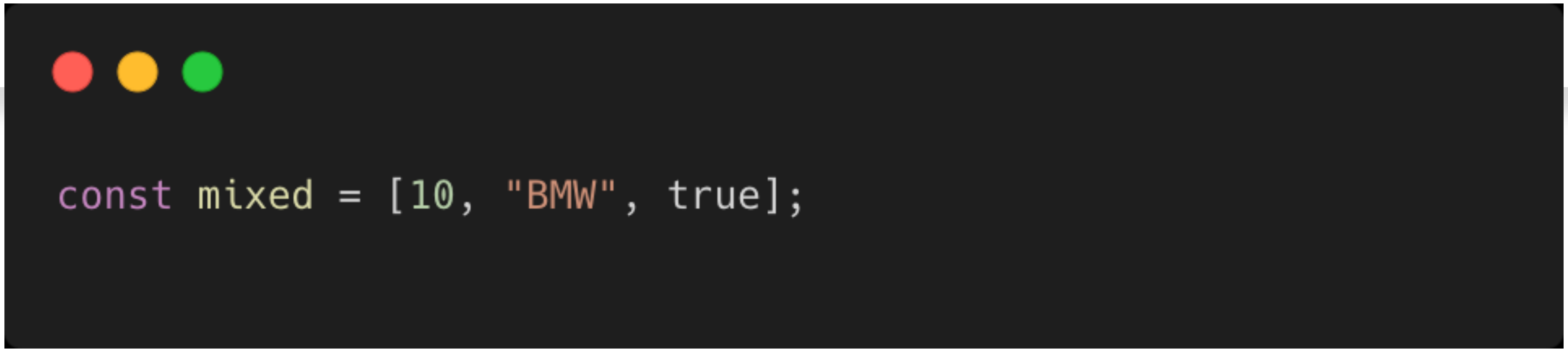
Each item in that array takes two blocks in the memory representation. When accessing these items from the memory, our program knows that the array is an array of Integers, so, it reads two bytes starting from the first and keeps on reading every next two bytes to complete the traversing.

If we have a mixed data types in an array in languages like Java or C++, for example, let's say it has 3 elements; first is an integer, second is a string and third is a Boolean:



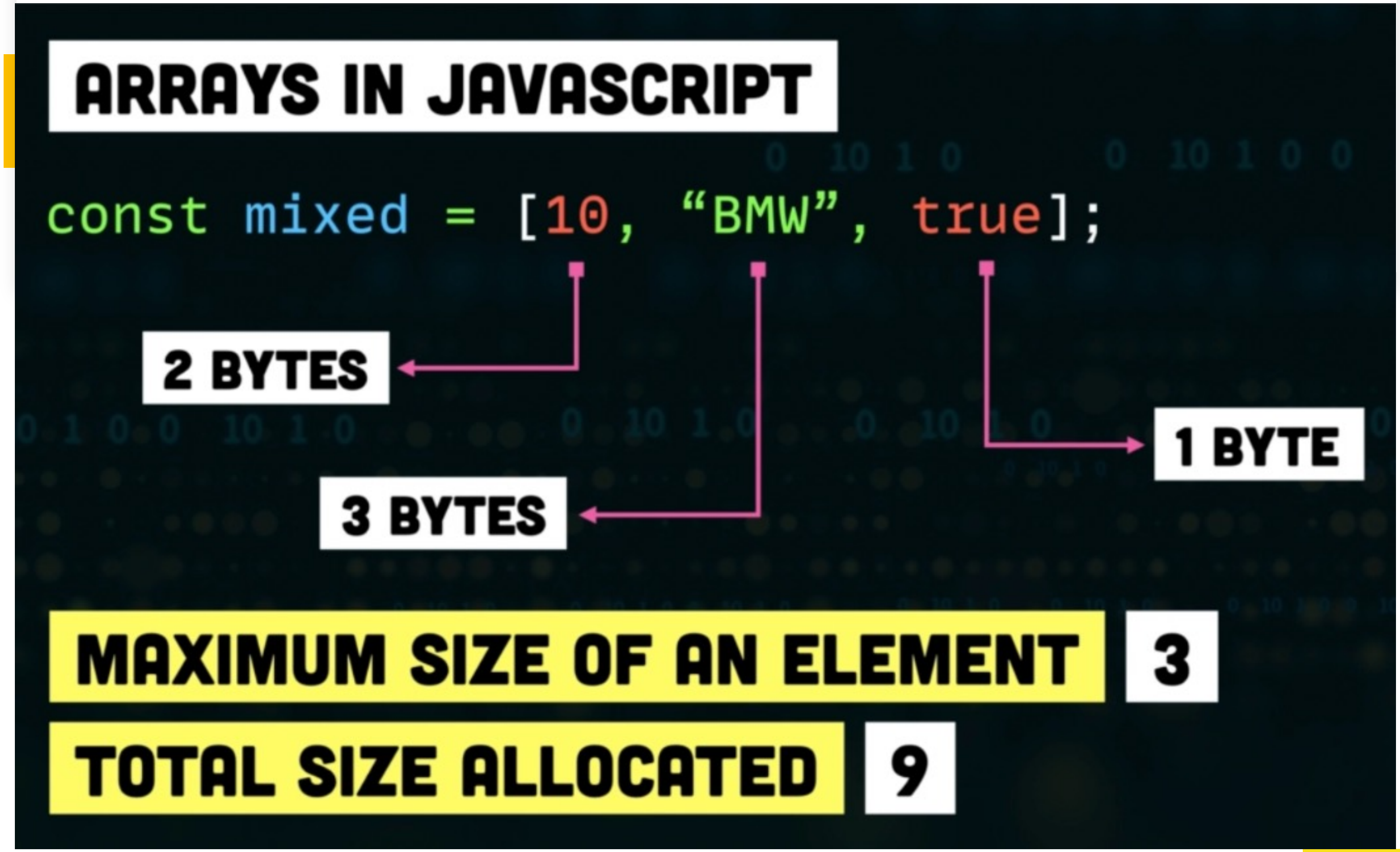
We have an integer which takes 2 bytes, a string which takes 3 bytes and a Boolean which takes 1 byte. When reading values from this array, since it is of mixed types, our program does not know how many bytes should it read to get the first, second and the third value.

However, in JavaScript, let's say, we have the same mixed array called "mixed".



```
const mixed = [10, "BMW", true];
```

We know that the integer takes 2 bytes of space in the memory, the string takes 3 bytes, and the Boolean value takes 1 byte. Let's see what happens in the memory:



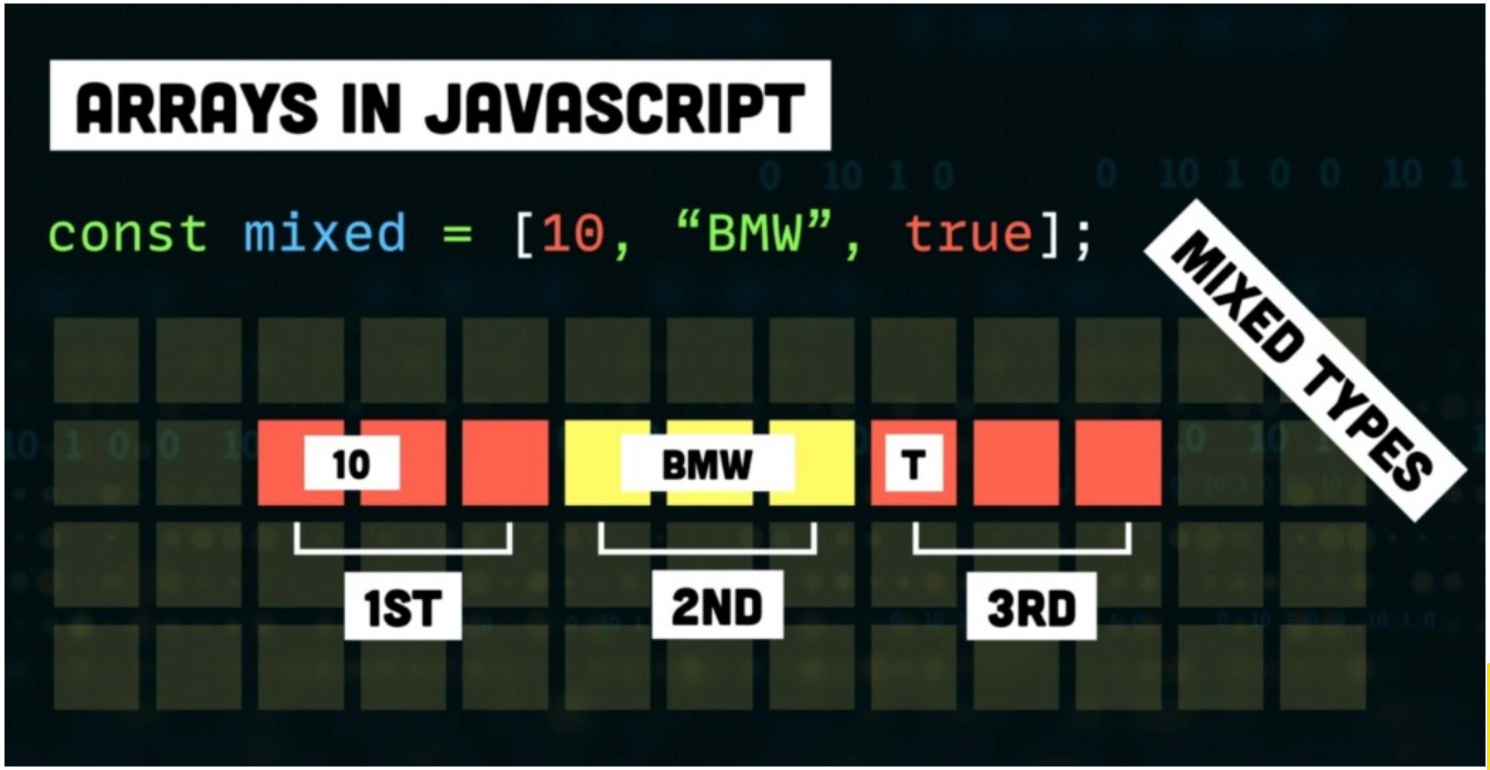
ARRAYS IN JAVASCRIPT

```
const mixed = [10, “BMW”, true];
```

MAXIMUM SIZE OF AN ELEMENT 3

TOTAL SIZE ALLOCATED 9

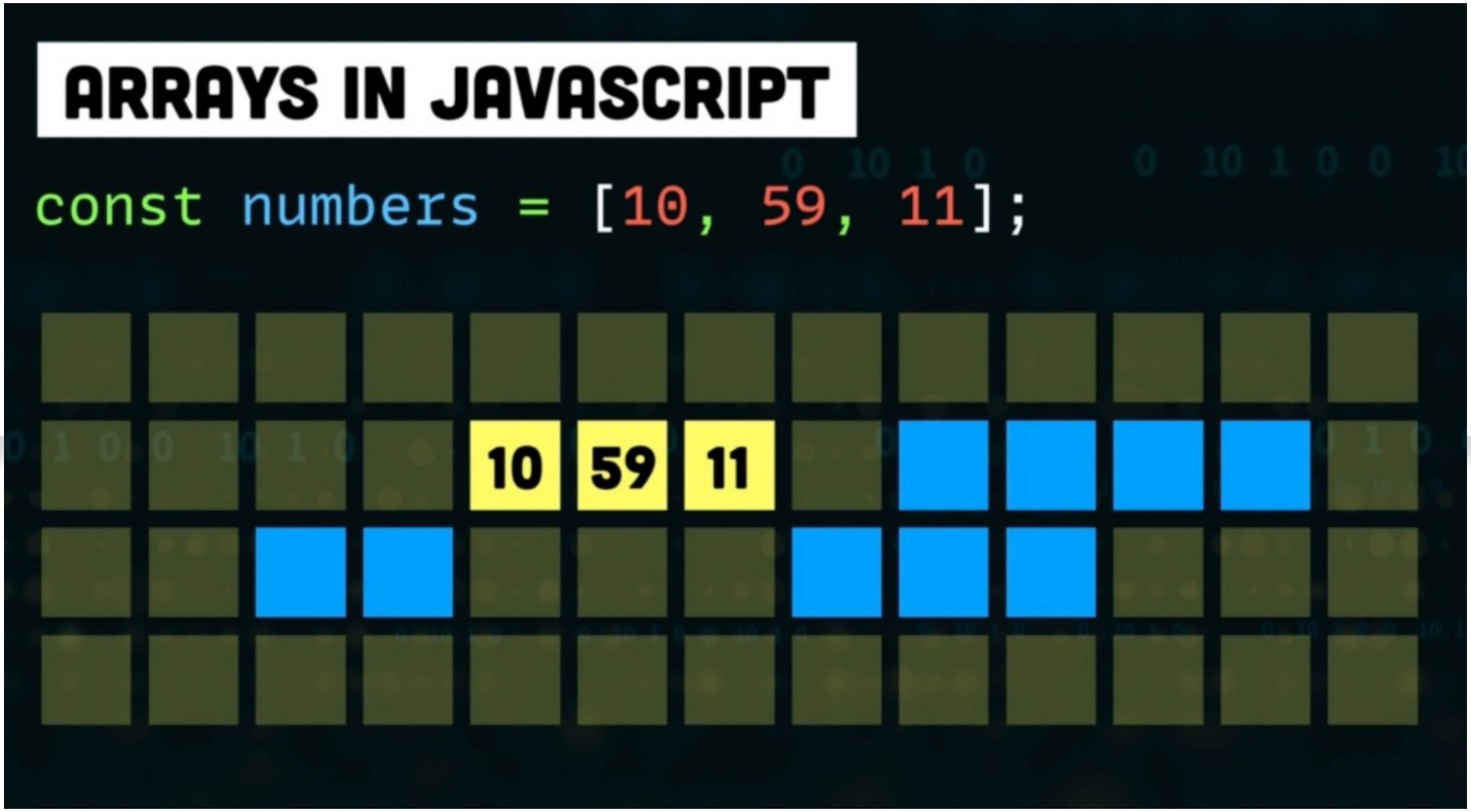
JS will find the maximum size of an element, in our case is the string "BMW" which take 3 bytes. And that size is allocated to each of the other elements in the array. Hence, JS will allocate a total of 9 bytes of memory for the 'mixed' array. This is called ‘boxing of size’ in JavaScript.



Why arrays in JavaScript can have dynamic size?

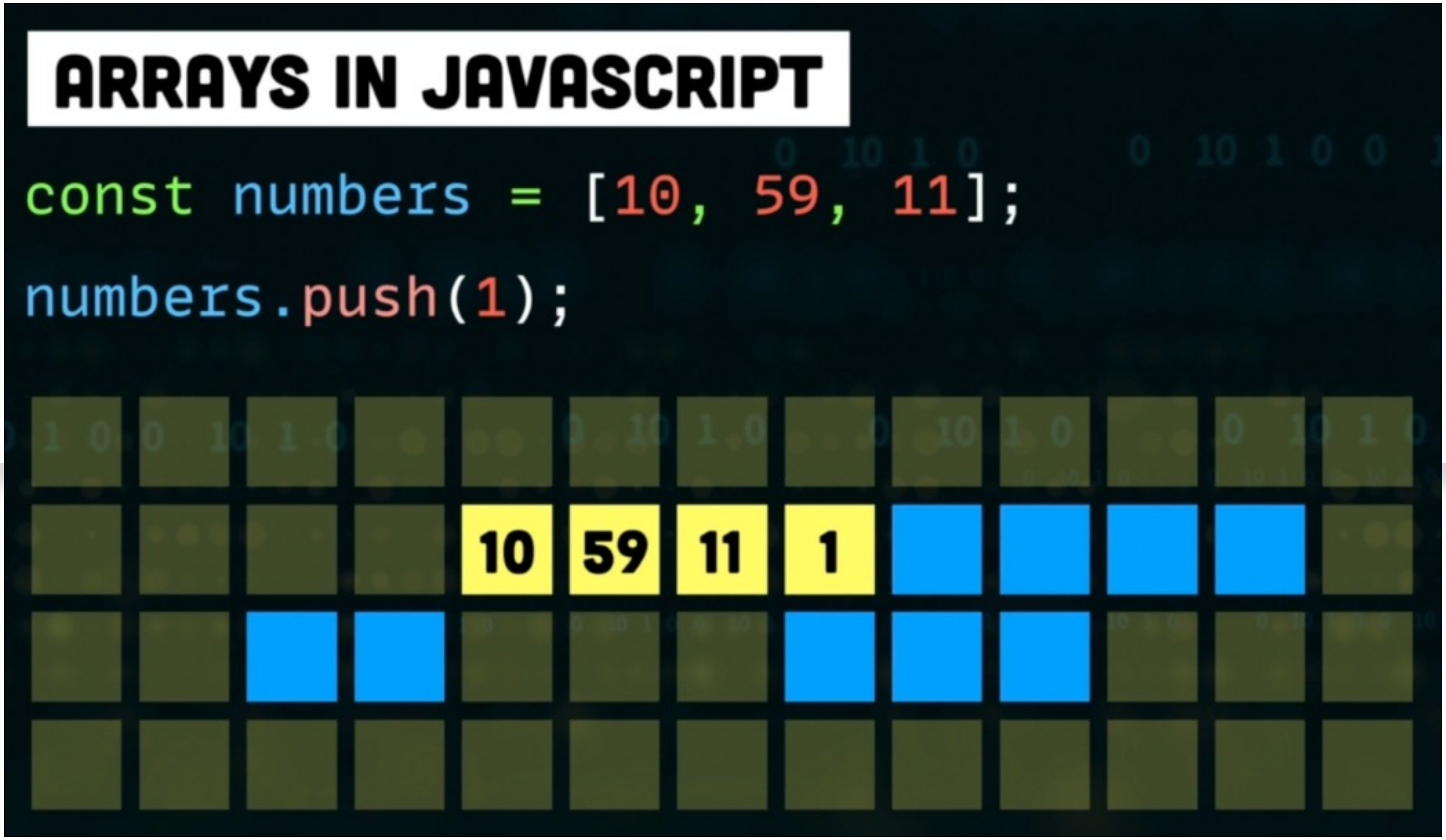
JavaScript arrays use dynamic memory allocation, which means that memory for array elements is allocated and managed dynamically by the JavaScript engine. When you add elements to an array, the engine takes care of allocating more memory if needed. Let’s see how is it done:

Let’s say we initialize an array in JavaScript, we call it “numbers” that holds 3 integers.



```
const numbers = [10, 59, 11];
```

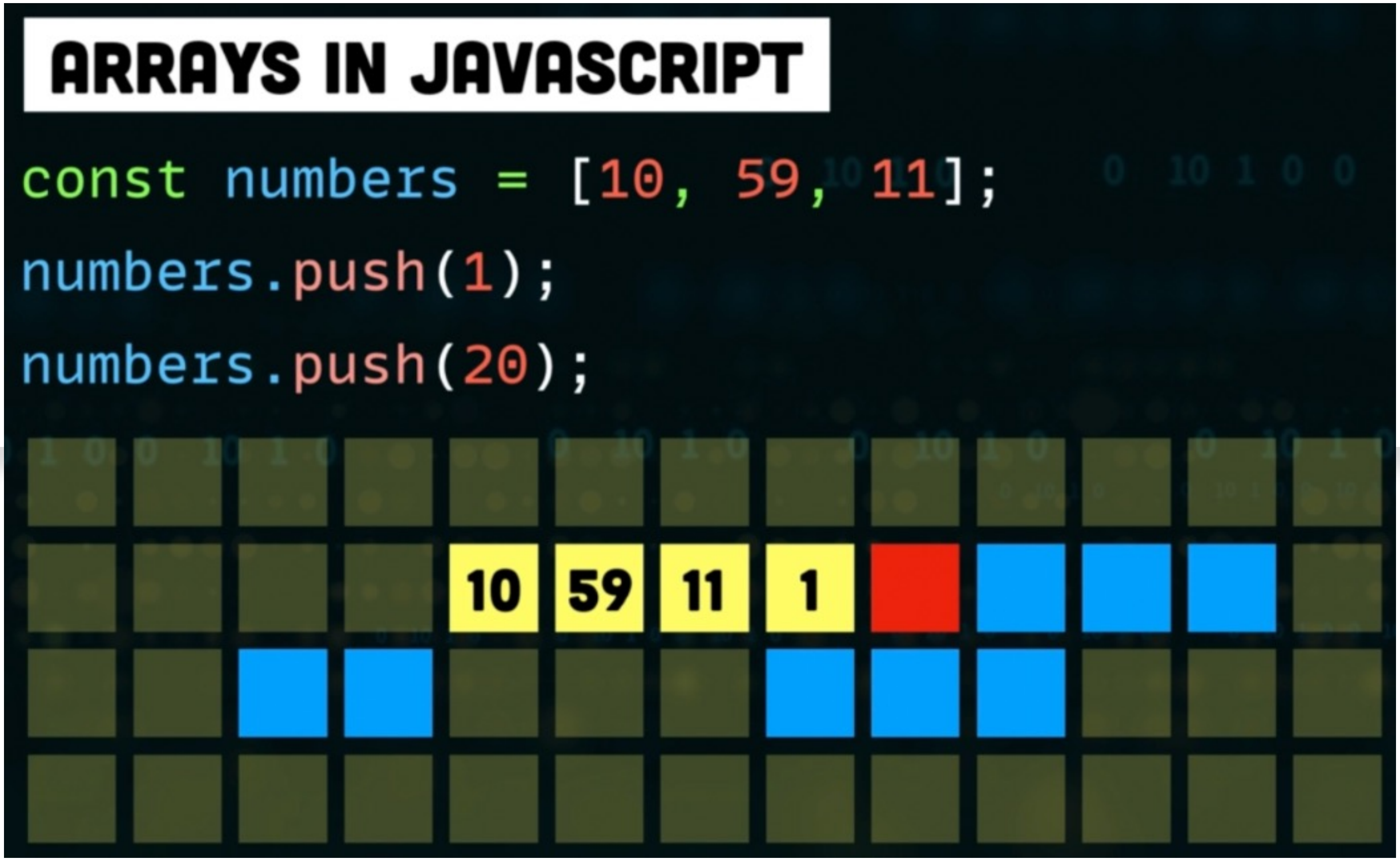
The 'numbers' array will be allocated in three blocks of the memory and other blocks can be used by other programs.



```
const numbers = [10, 59, 11];

numbers.push(1);
```

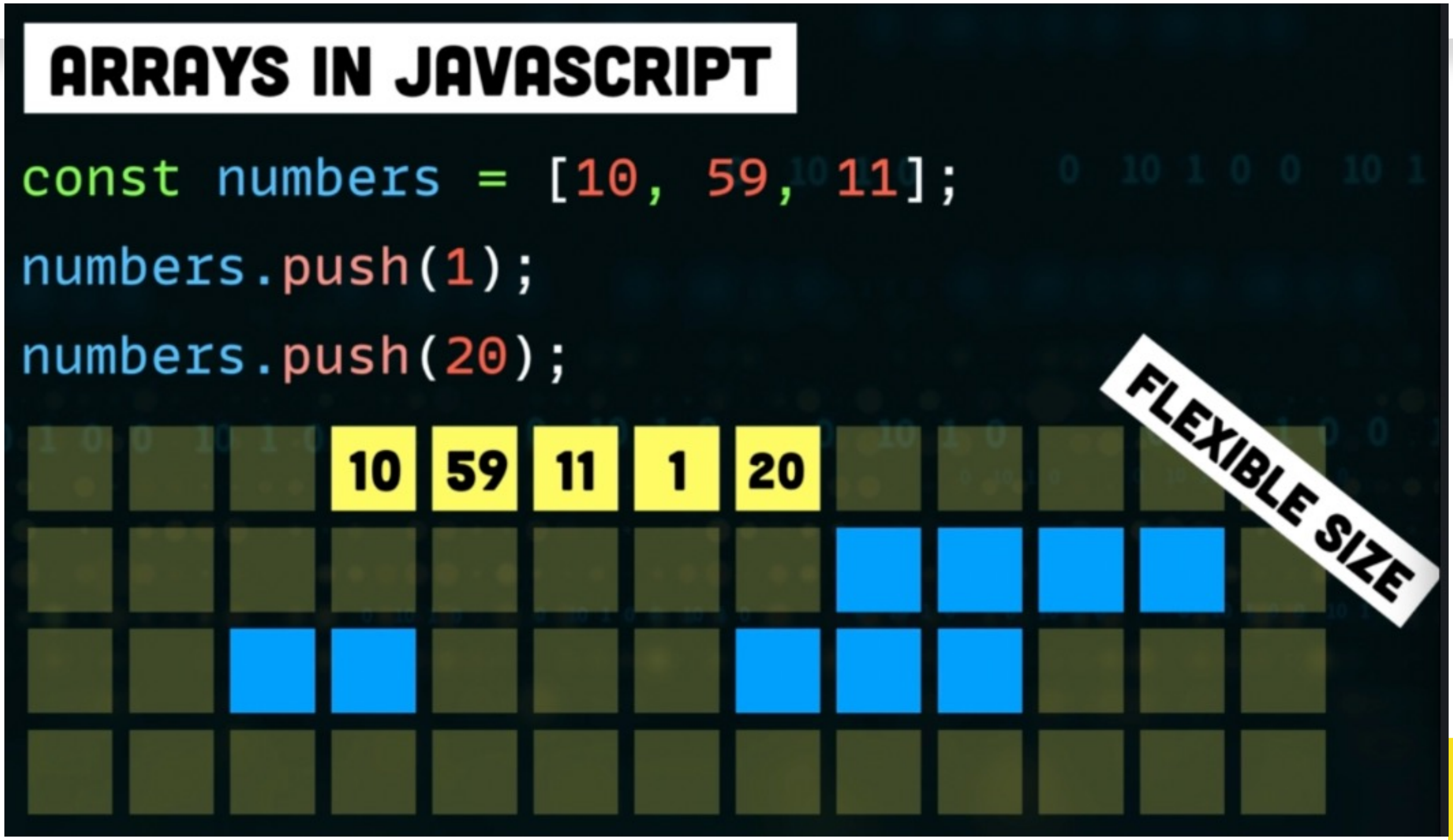
When we pushed a new item to the 'numbers' array, the program checks if there is a space for the item and if yes, it will allocate that space for the new item.



```
const numbers = [10, 59, 11];
numbers.push(1);
numbers.push(20);
```

If we try to push a new number, it cannot be consecutively added to the memory as the block is already occupied by another program.

In that case, the program tries to find a whole new space in the memory where it can consecutively populate the space with all the items , moves all the items to the space, frees up the previous space and pushes the new item to the new consecutive space;


ARRAYS IN JAVASCRIPT

```
const numbers = [10, 59, 11];
numbers.push(1);
numbers.push(20);
```

This is why arrays in JavaScript are dynamic.

THANK YOU !!

Please comment and share.