Deep Copy
Shallow copy

To know ->scroll through

First of all, what is a copy?

A copy just looks like the old thing, but isn't. When you change the copy, you expect the original thing to stay the same, whereas the copy changes.

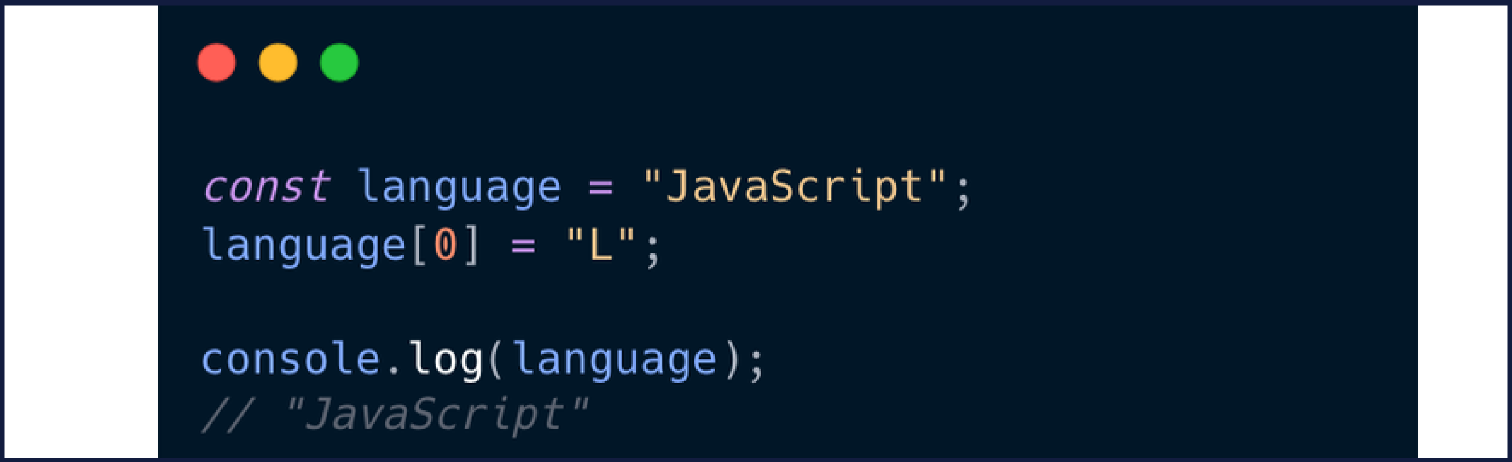
The first thing worth explaining, even before moving on to copies, are data values in JavaScript.

- There are 8 data types which belong to two different value types: primitives and objects. Let's see how the two differ.

Primitives are all the data types except objects. That means:

- Boolean
- Null
- Undefined
- String
- Number
- Bigint
- Symbol

Primitives are immutable

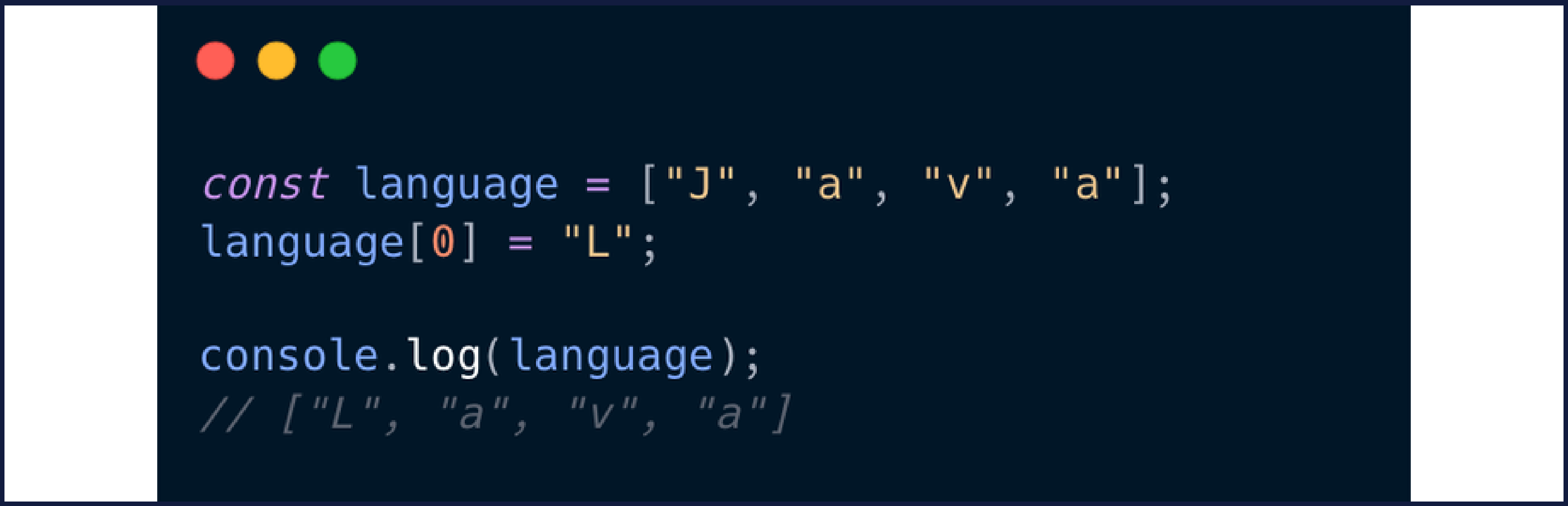


```javascript
const language = "JavaScript";
language[0] = "L";

console.log(language);
// "JavaScript"
```

When attempting to change the first letter of a string "J" to "L," we receive the language variable not altered. As its a immutable property

Objects are all other JavaScript elements, such as object literals, arrays, dates, etc.,

- **Objects are mutable**



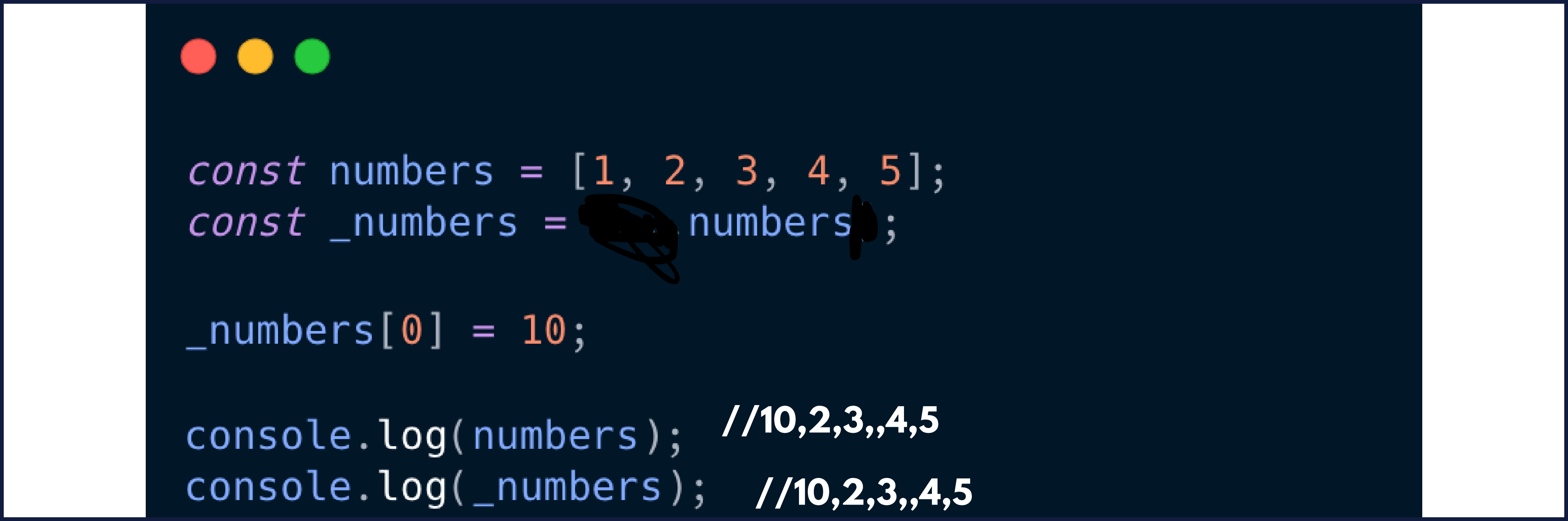
```
const language = ["J", "a", "v", "a"];
language[0] = "L";

console.log(language);
// ["L", "a", "v", "a"]
```

Here we can see that changing the first element of the array resulted in mutating the language variable.

We have two kinds of object copies in JavaScript:

- **shallow and**
- **deep**

In Code we will see the problem as its changing the original array



```
const numbers = [1, 2, 3, 4, 5];
const _numbers = [redacted] numbers[redacted];

_numbers[0] = 10;

console.log(numbers);  //10,2,3,,4,5
console.log(_numbers);  //10,2,3,,4,5
```

To get ride of this problem we have

- **Shallow copy**
- **Deep copy**

Deep copying vs. Shallow copying

- A deep copy means that all of the values of the new variable are copied and disconnected from the original variable.
- A shallow copy means that certain (sub-)values are still connected to the original variable.

In a nutshell,

- shallow copies are used for “flat” objects
- deep copies are used for “nested” object

By **“flat” objects** we mean objects that contain only primitive values.

For instance: [1, 2, 3, 4, 5]

Nested objects mean objects that contain non-primitive values.

For instance: [“laptop”, {value: 5000}]

To create a shallow copy, we can use the following methods:

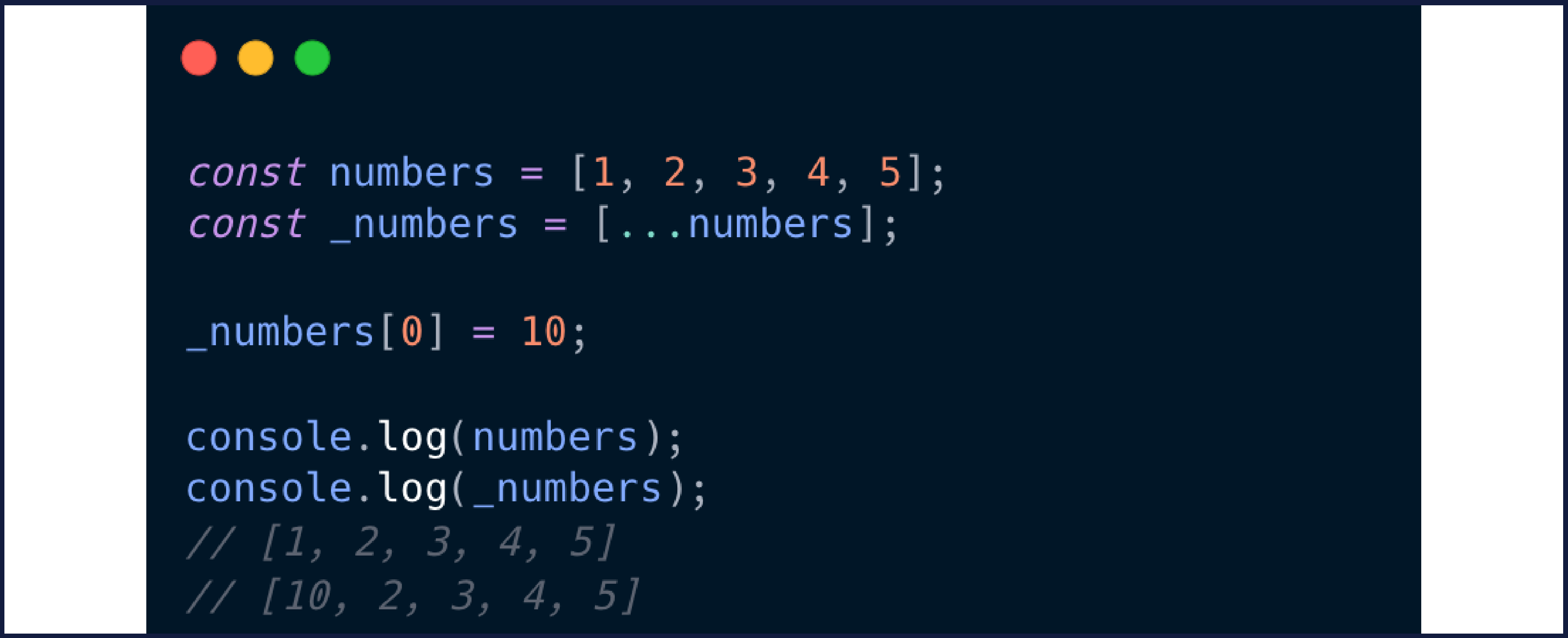
- Spread syntax [...] {...}
- Object.assign()
- Array.from()
- Object.create()
- Array.prototype.concat()

And to create a deep copy, we can use:

- JSON.parse(JSON.stringify())
- structuredClone()
- Third party libraries like Lodash

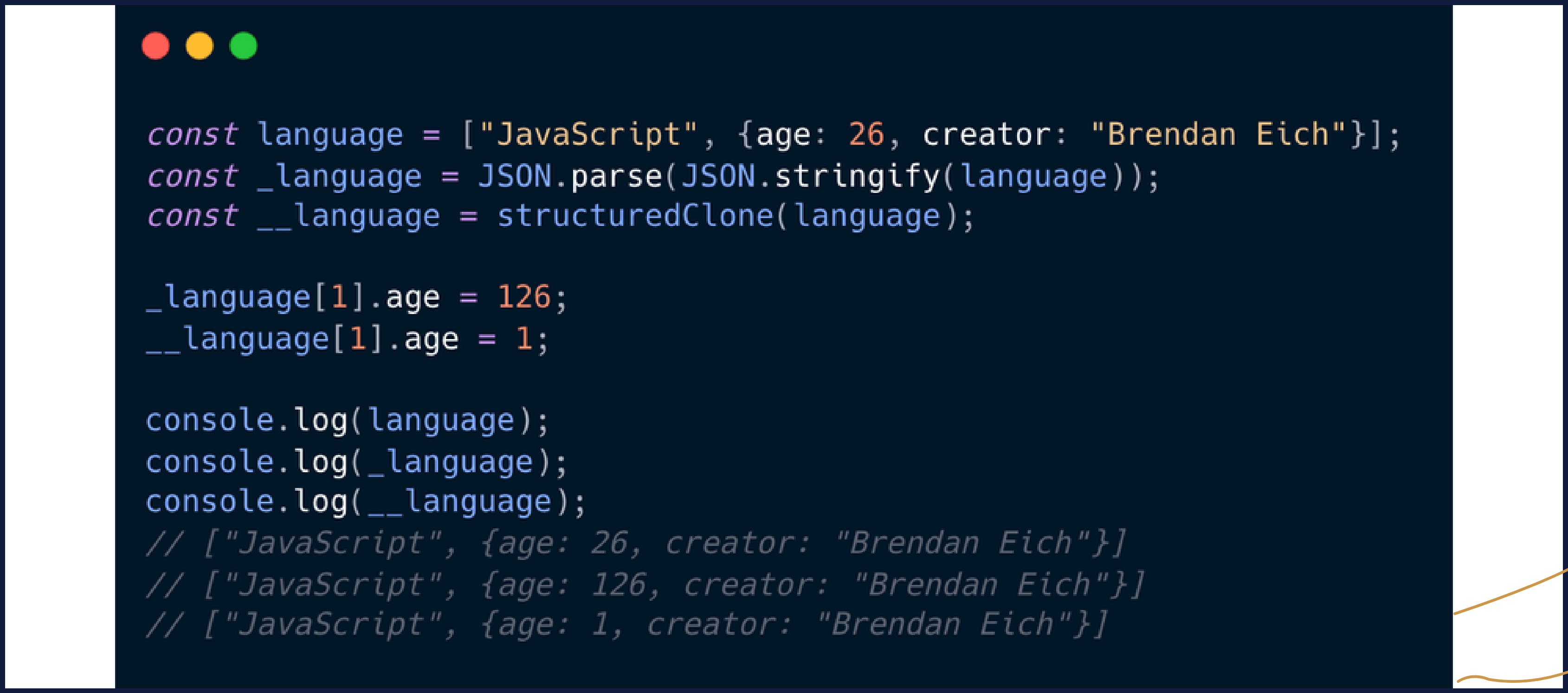
I will give a one example of shallow copy and deep copy method example

Shallow copy: Spread operator method



```javascript
const numbers = [1, 2, 3, 4, 5];
const _numbers = [...numbers];

_numbers[0] = 10;

console.log(numbers);
console.log(_numbers);
// [1, 2, 3, 4, 5]
// [10, 2, 3, 4, 5]
```

Deep copy: JSON.stringify() method



```javascript
const language = ["JavaScript", {age: 26, creator: "Brendan Eich"}];
const _language = JSON.parse(JSON.stringify(language));
const __language = structuredClone(language);

_language[1].age = 126;
__language[1].age = 1;

console.log(language);
console.log(_language);
console.log(__language);
// ["JavaScript", {age: 26, creator: "Brendan Eich"}]
// ["JavaScript", {age: 126, creator: "Brendan Eich"}]
// ["JavaScript", {age: 1, creator: "Brendan Eich"}]
```