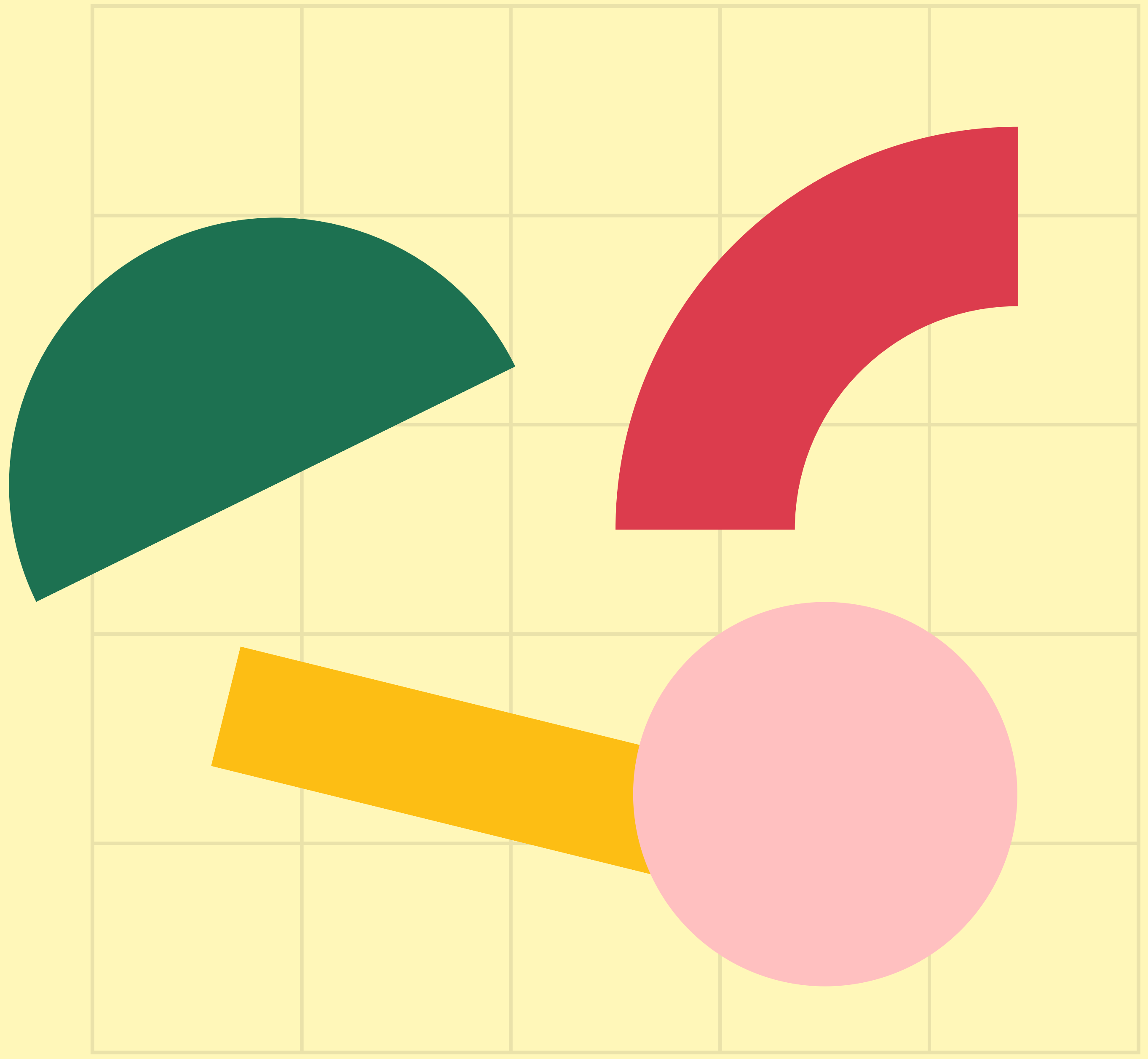
JavaScript Basics

Course for Beginners

Unit Goals

what we'll cover

- primitive types
- running code in the console
- numbers
- math operations
- variables
- basic syntax
- recall values
- const, var
- booleans

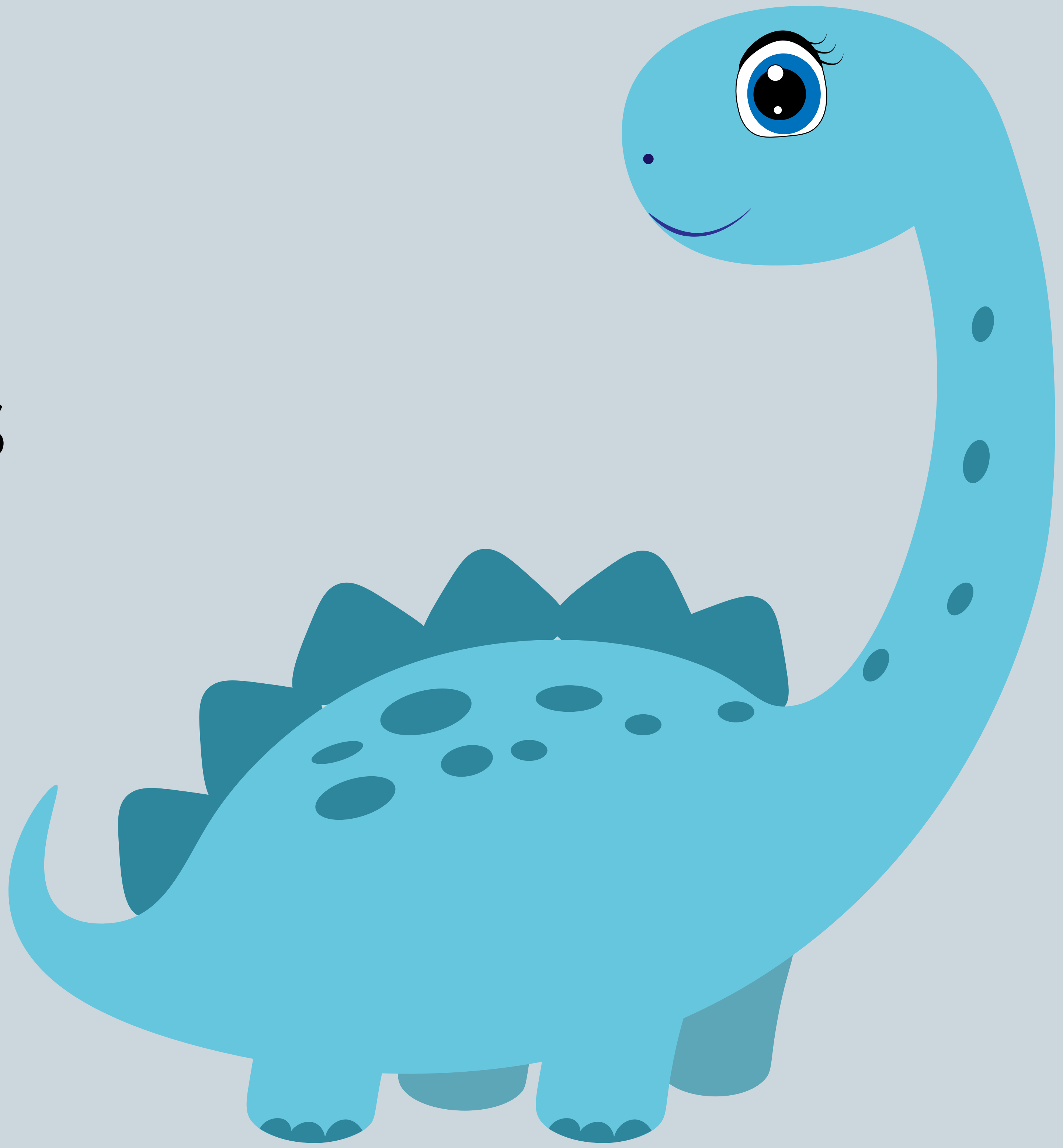


THE

BLUE ----------- CSS - adjectives

DINO ----------- HTML - nouns

SMILED ---- JS - verbs

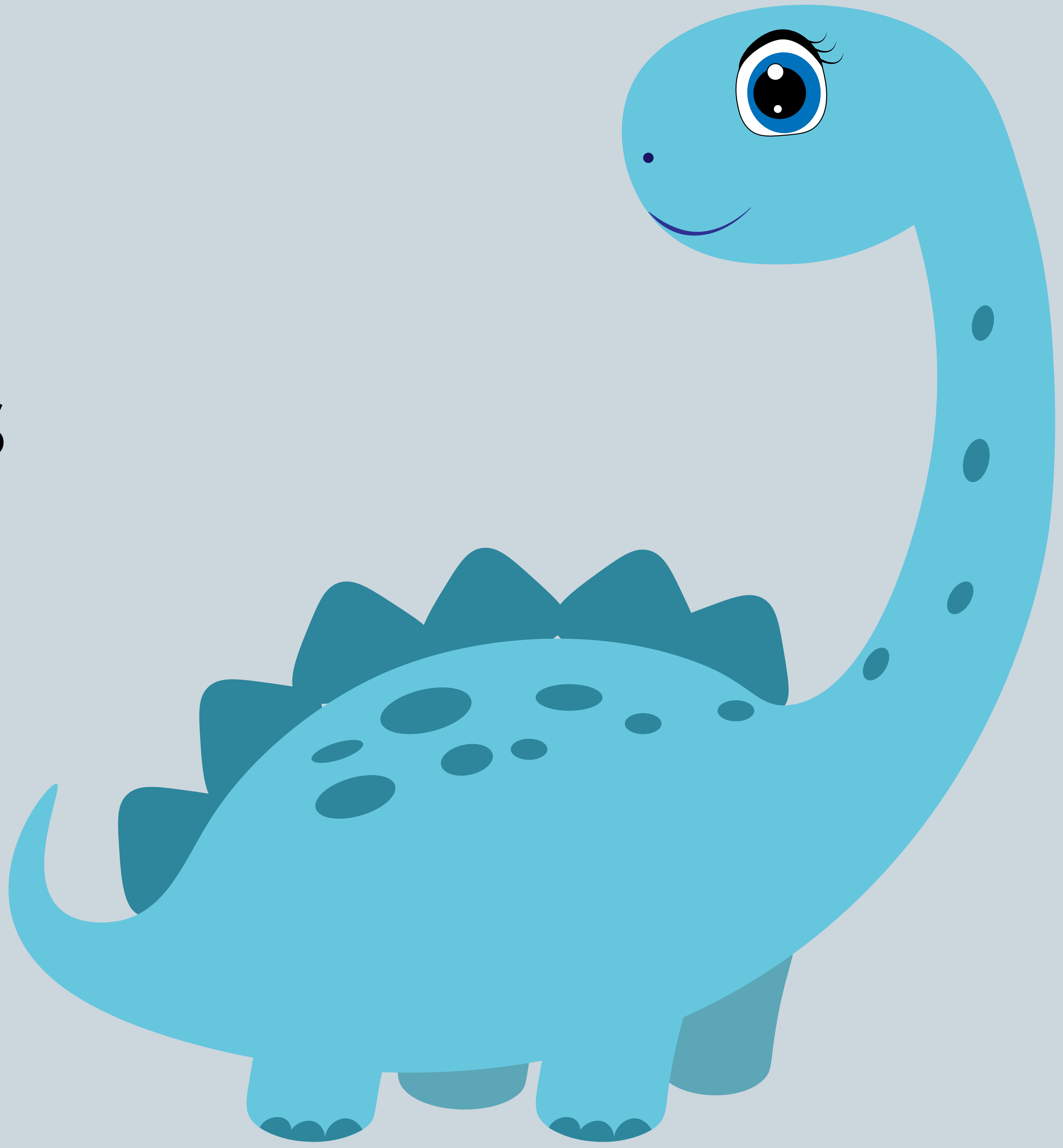


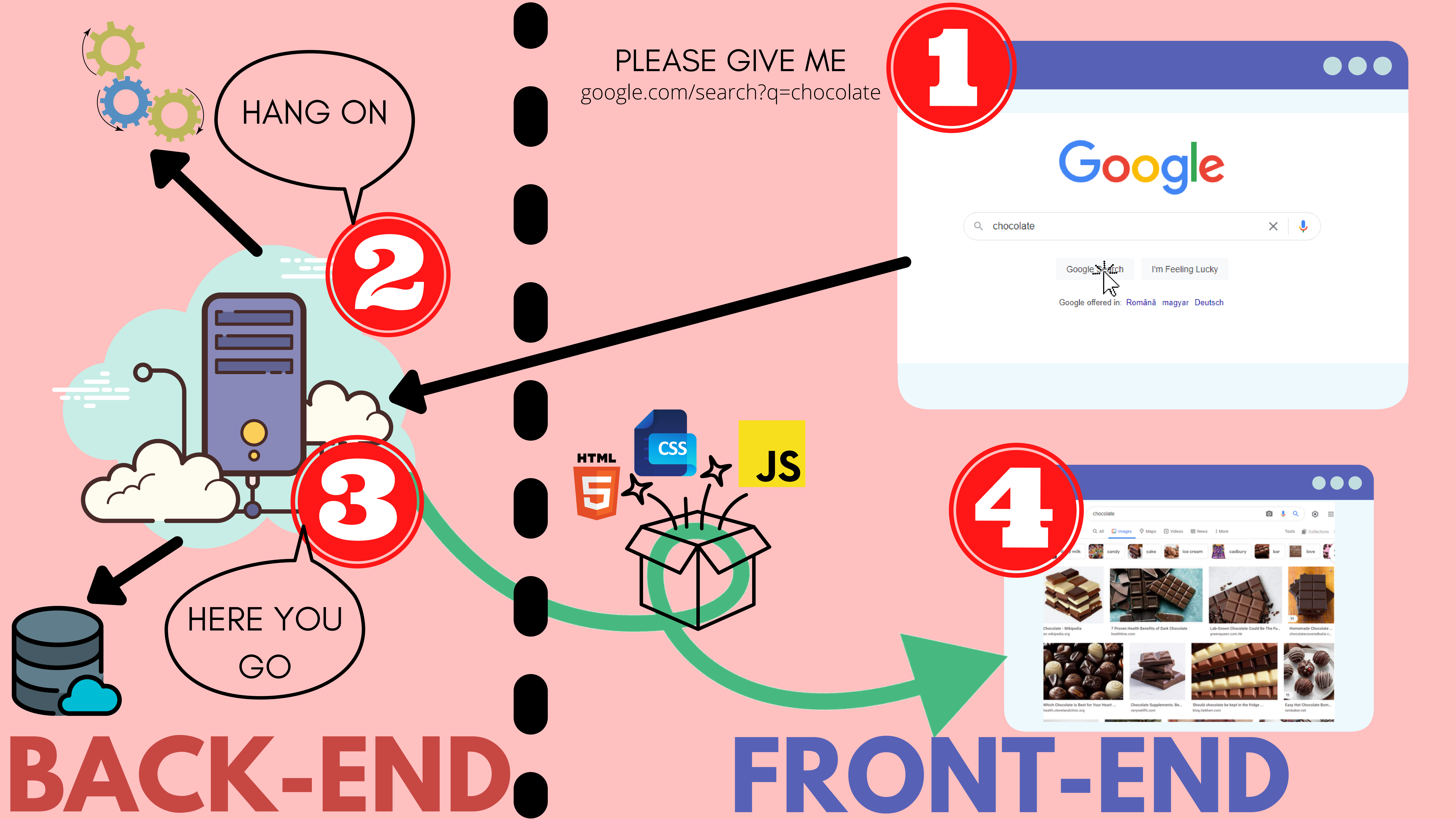
THE

BLUE ----------- CSS - adjectives

DINO ----------- HTML - nouns

SMILED ---- JS - verbs




PLEASE GIVE ME
google.com/search?q=chocolate
1
Google
chocolate
Google Search
I'm Feeling Lucky
Google offered in: Română magyar Deutsch
2
HANG ON
3
HERE YOU GO
HTML
CSS
JS
4
chocolate
BACK-END
FRONT-END

HTML
CSS
JS

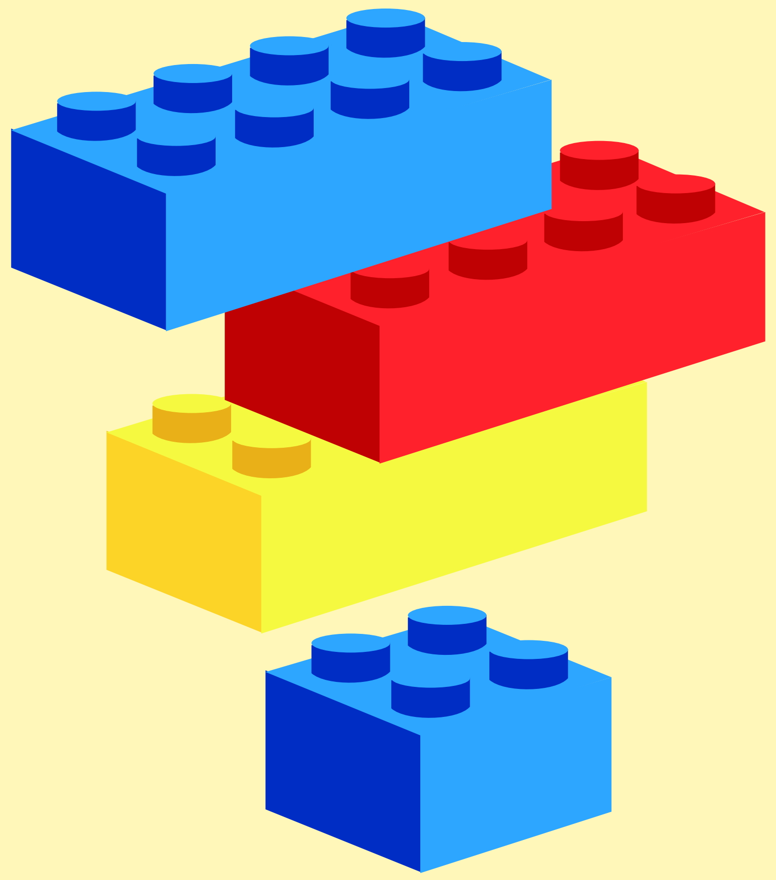
1 LEARN JS ON ITS OWN - NO HTML/CSS

2 USE JS TO MANIPULATE HTML/CSS

PRIMITIVE TYPES

The basic building blocks*:

- Number
- String
- Boolean
- Null
- Undefined

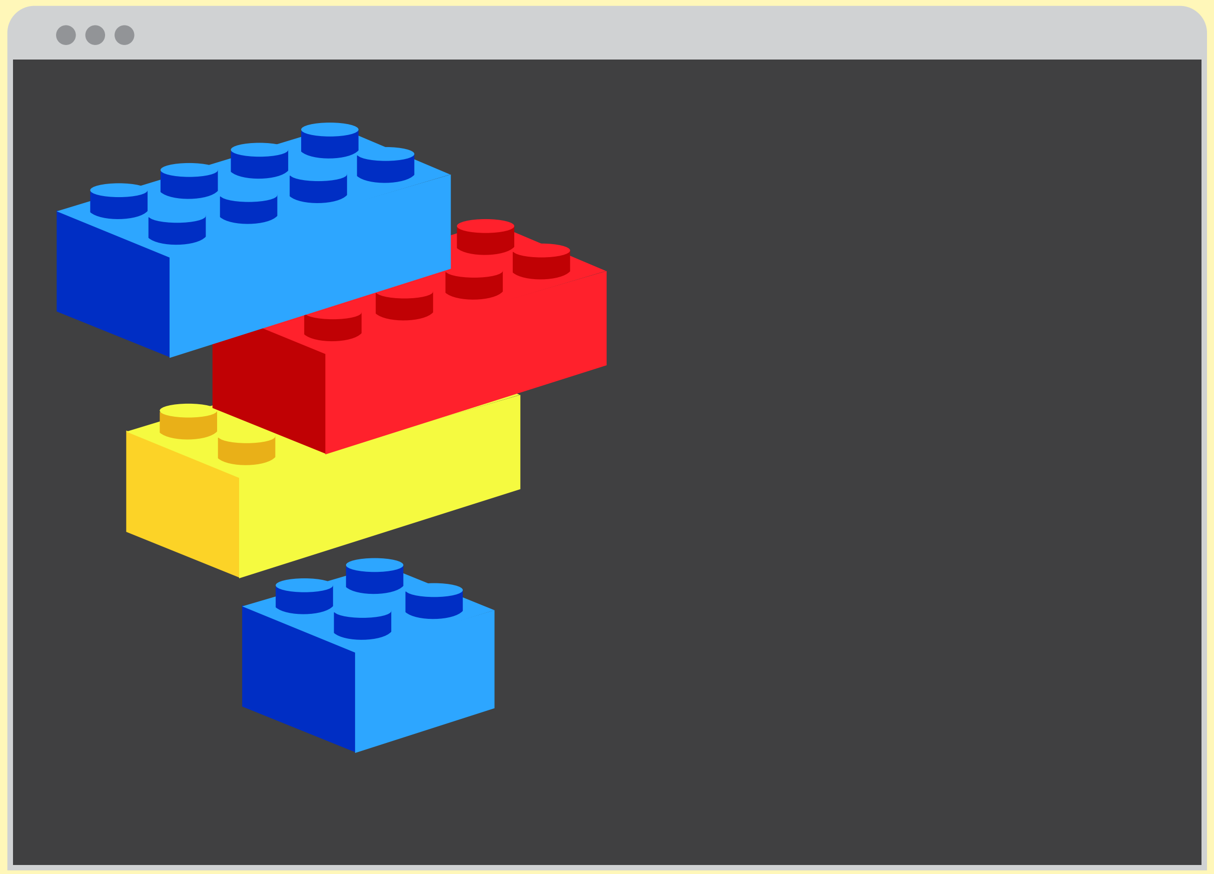


*technically there are two more others: Symbol and BigInt

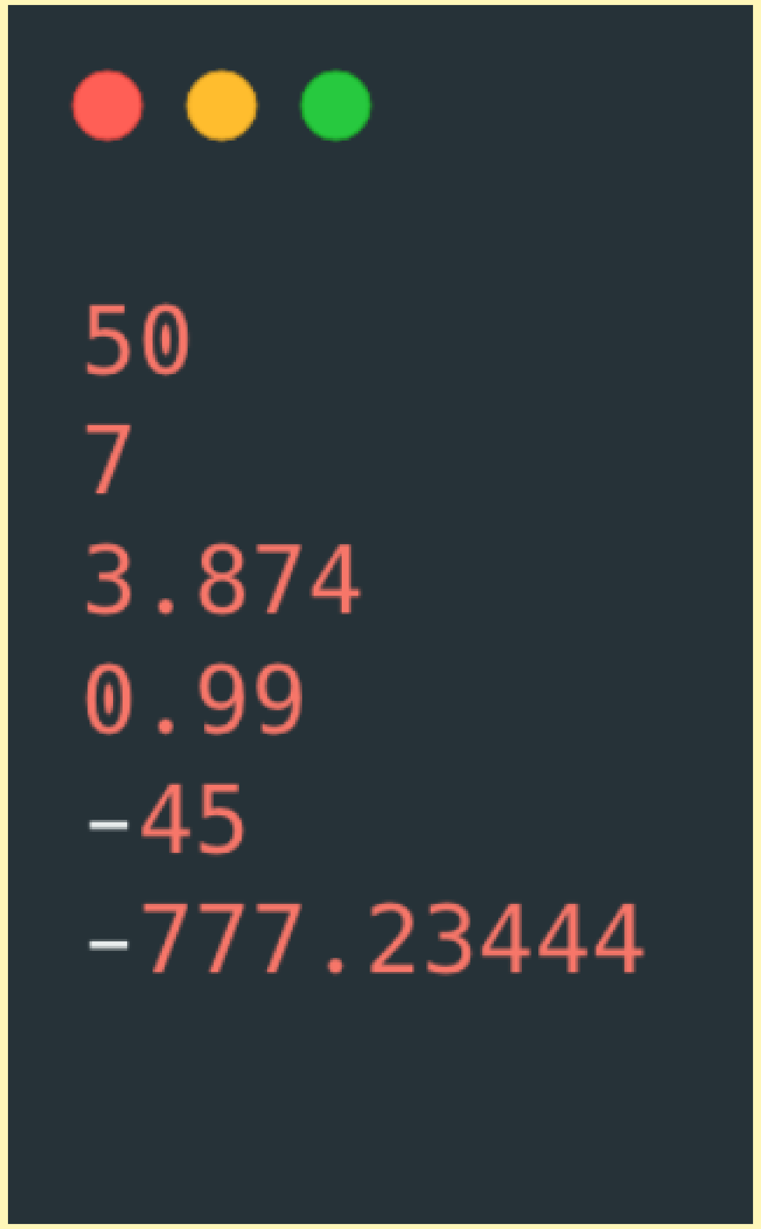
RUNNING CODE IN THE CONSOLE

THE EASIEST PLACE TO START

Early on, we'll run our code using the Chrome developer tools console. Then, we'll learn how to write external scripts.



NUMBERS

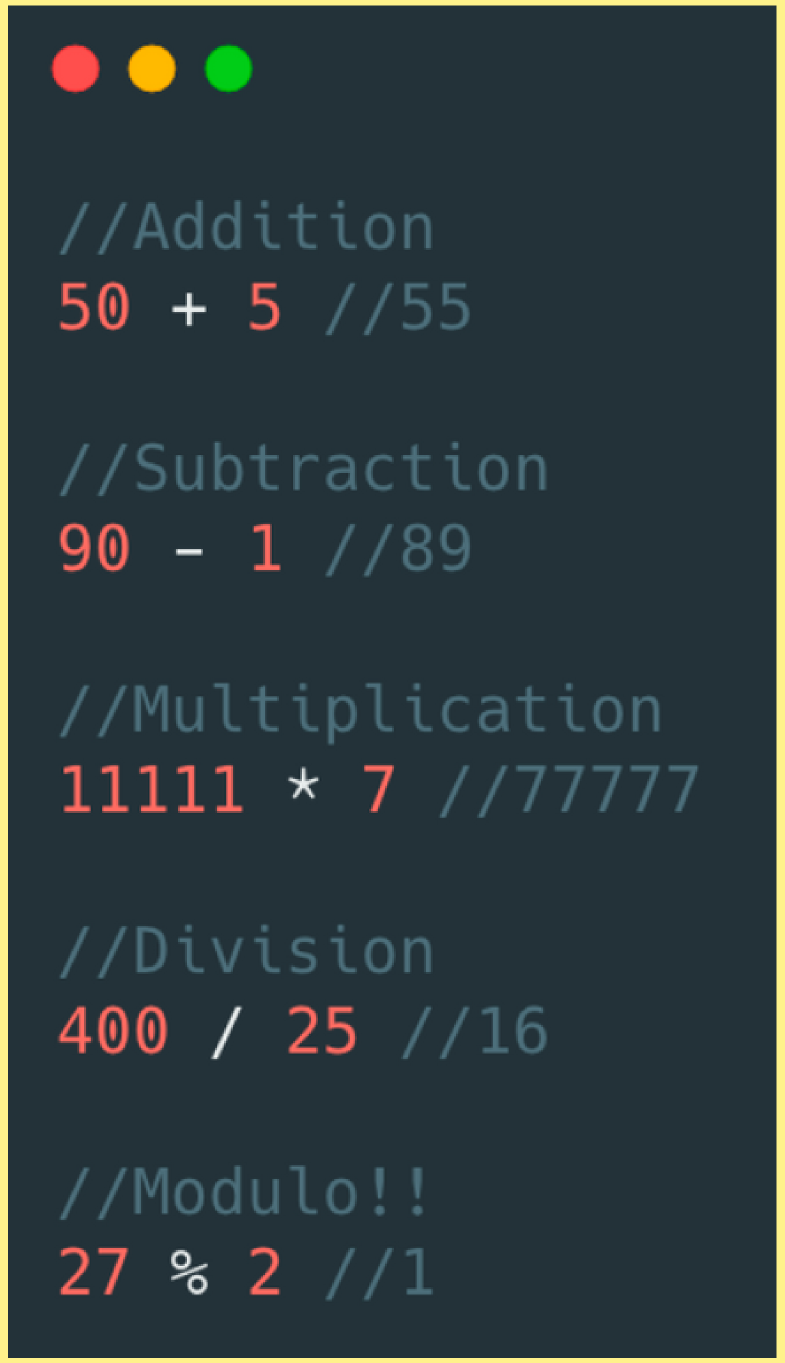


```
50
7
3.874
0.99
-45
-777.23444
```

IN JAVASCRIPT

- JS has one number type
    - Positive numbers
    - Negatives numbers
    - Whole numbers (integers)
    - Decimal numbers

MATH OPERATIONS



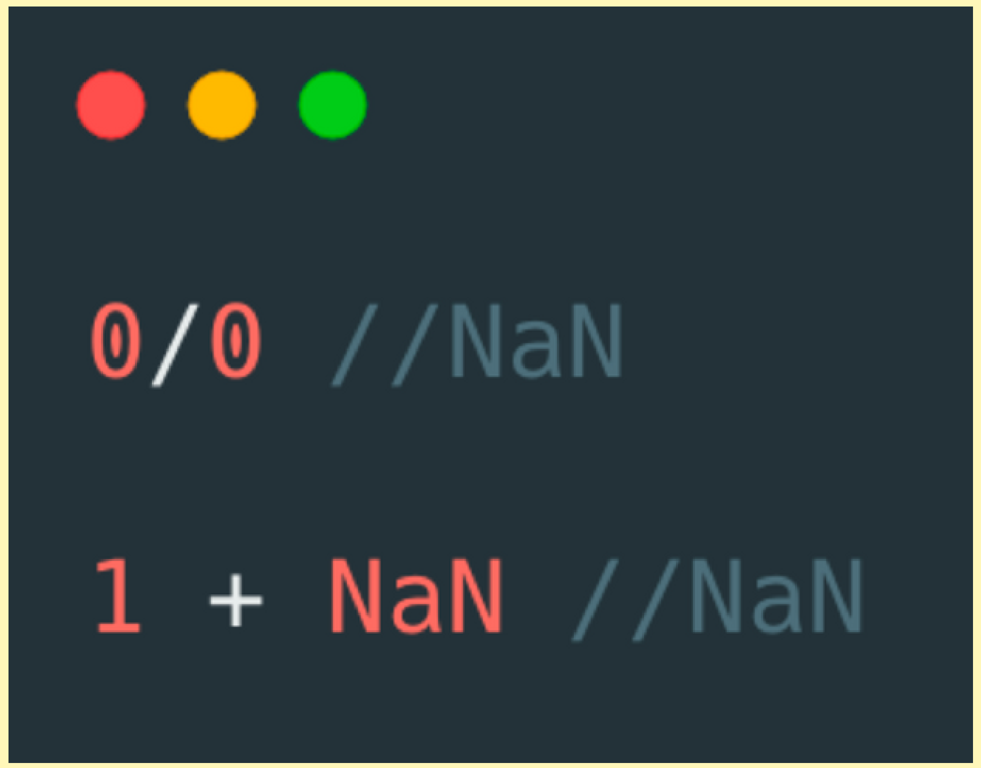
```
//Addition
50 + 5 //55

//Subtraction
90 - 1 //89

//Multiplication
11111 * 7 //77777

//Division
400 / 25 //16

//Modulo!!
27 % 2 //1
```

//creates a comment
//(the line is ignored)

NaN

Not a Number

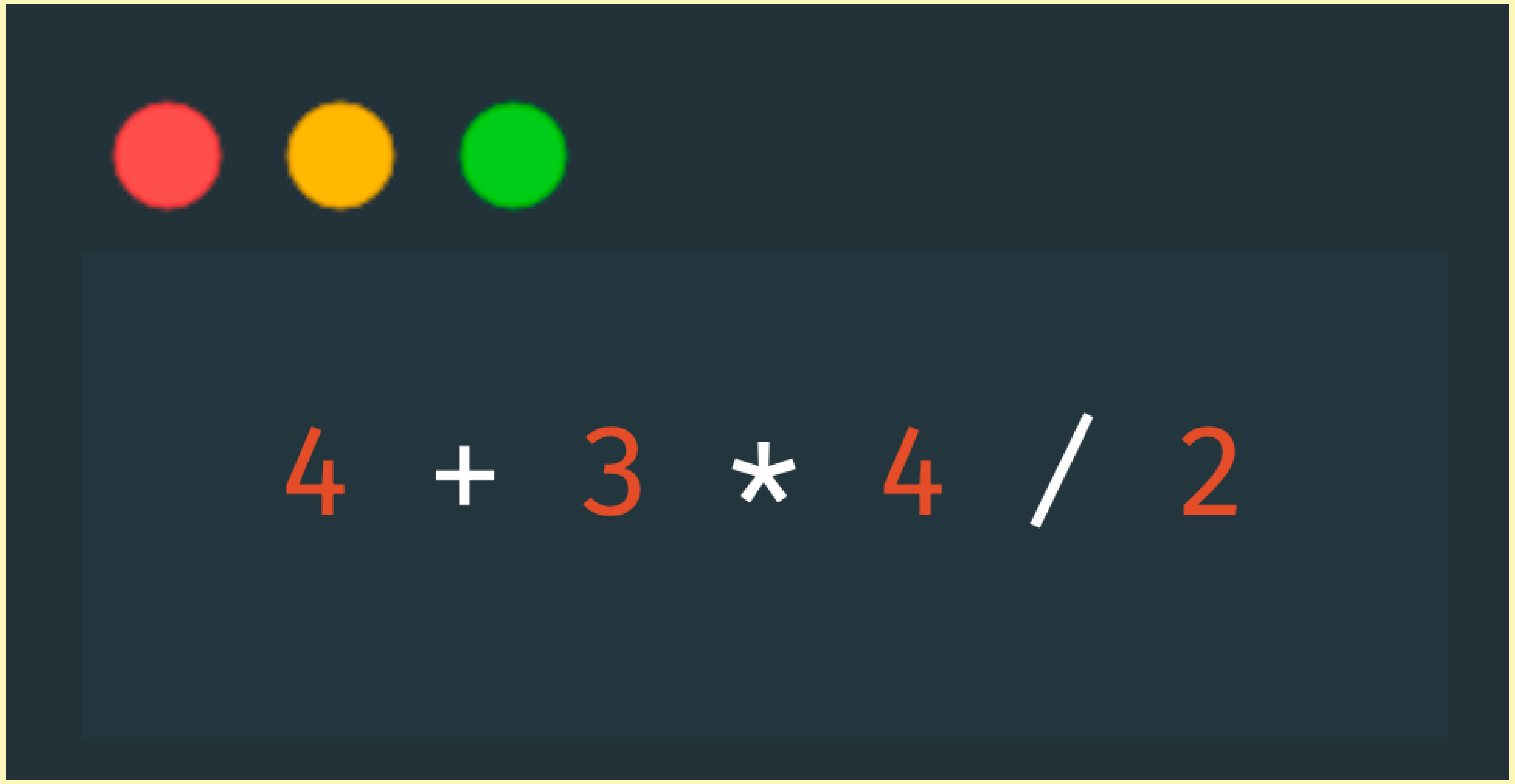
NaN is a numeric value that represents something that is not a number.



```
0/0 //NaN

1 + NaN //NaN
```

EVALUATION ORDER

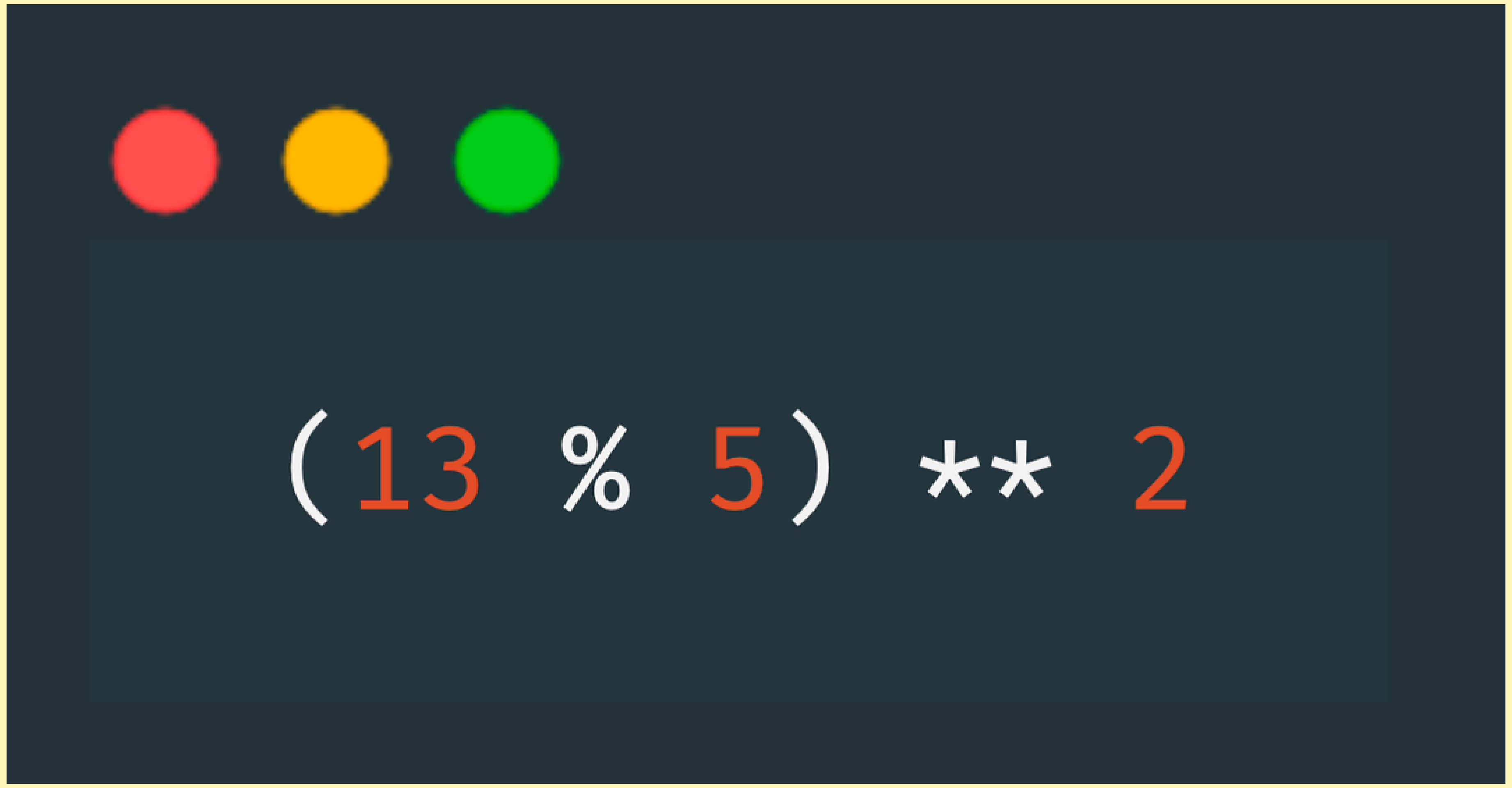
WHAT DOES THIS EVALUATE TO?



```
4 + 3 * 4 / 2
```

EVALUATION ORDER

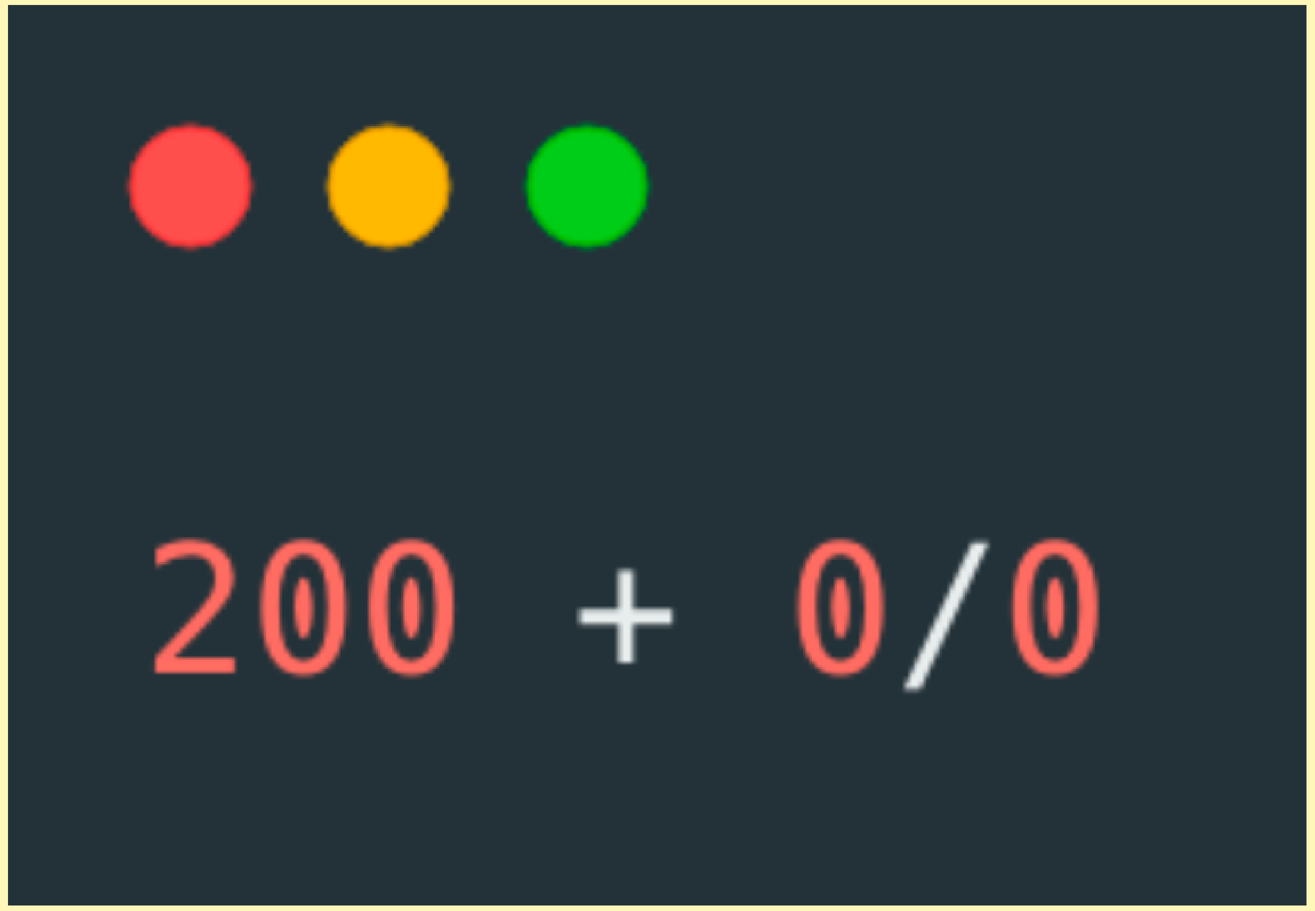
WHAT DOES THIS EVALUATE TO?



```
(13 % 5) ** 2
```

EVALUATION ORDER

WHAT DOES THIS EVALUATE TO?



```
200 + 0/0
```

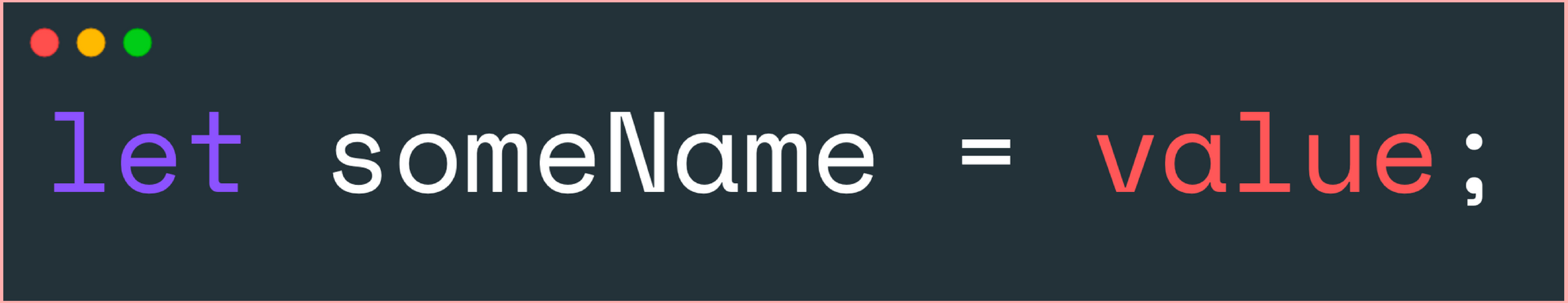
VARIABLES

VARIABLES ARE LIKE VALUES FOR VALUES

We can store a value and give it a name so that we can:

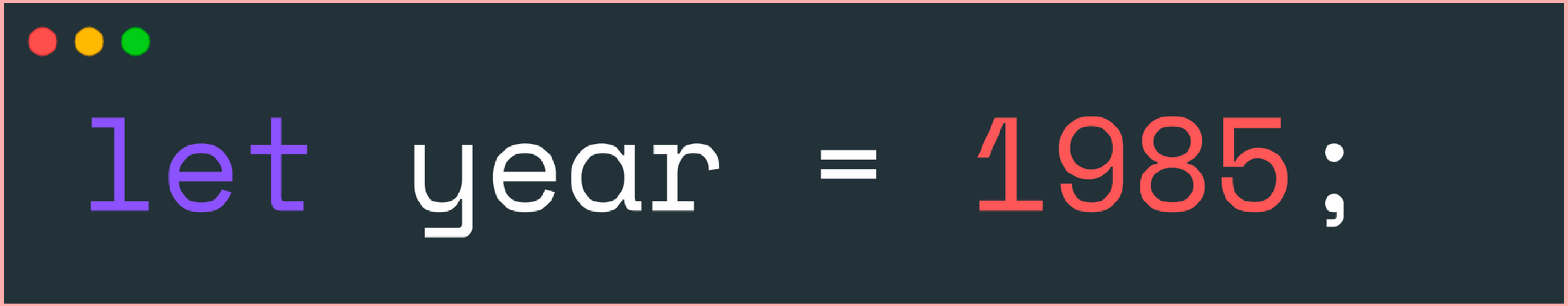
- refer back to it later
- use that value to do stuff
- change it later one

BASIC SYNTAX



```
let someName = value;
```

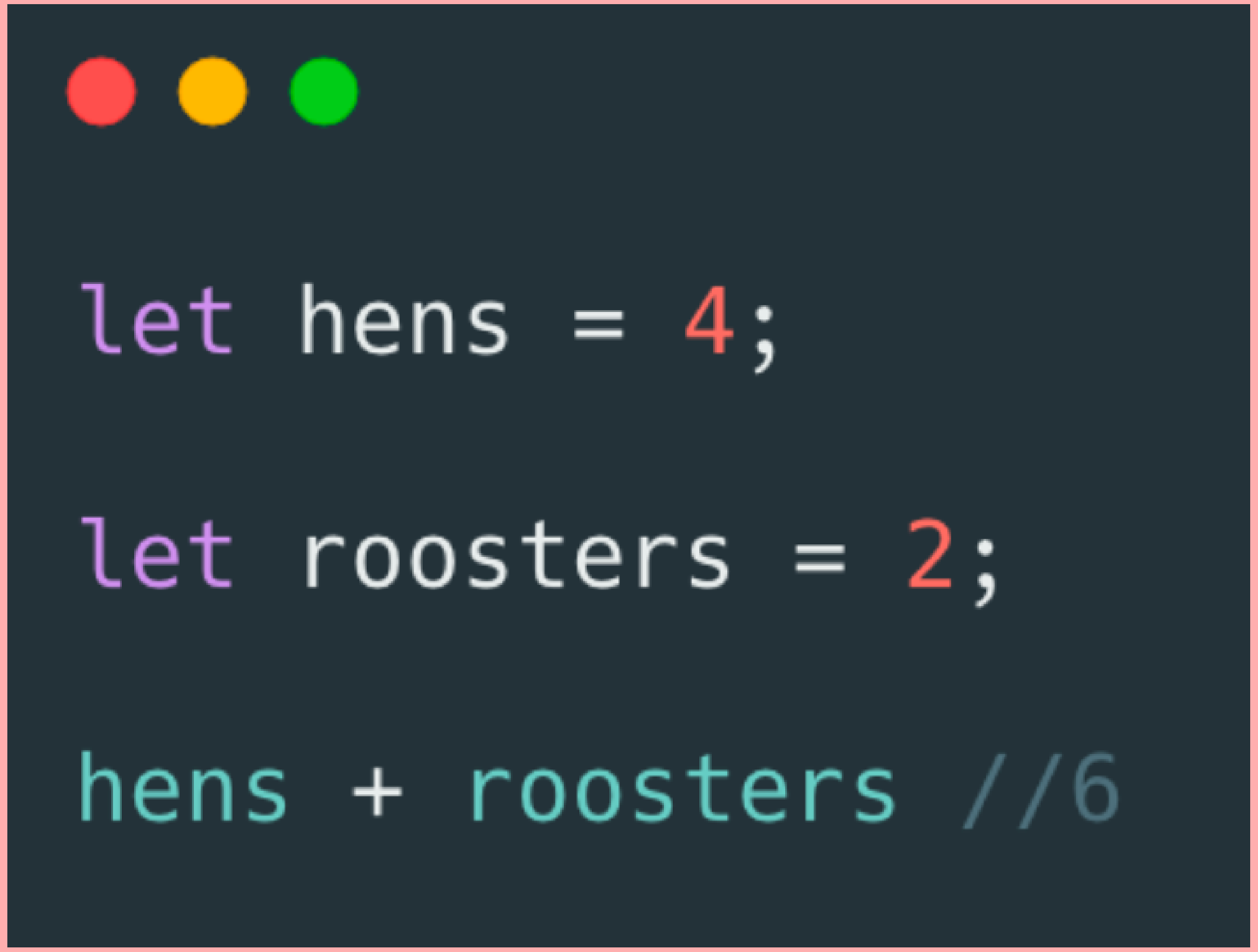
BASIC SYNTAX



```
let year = 1985;
```

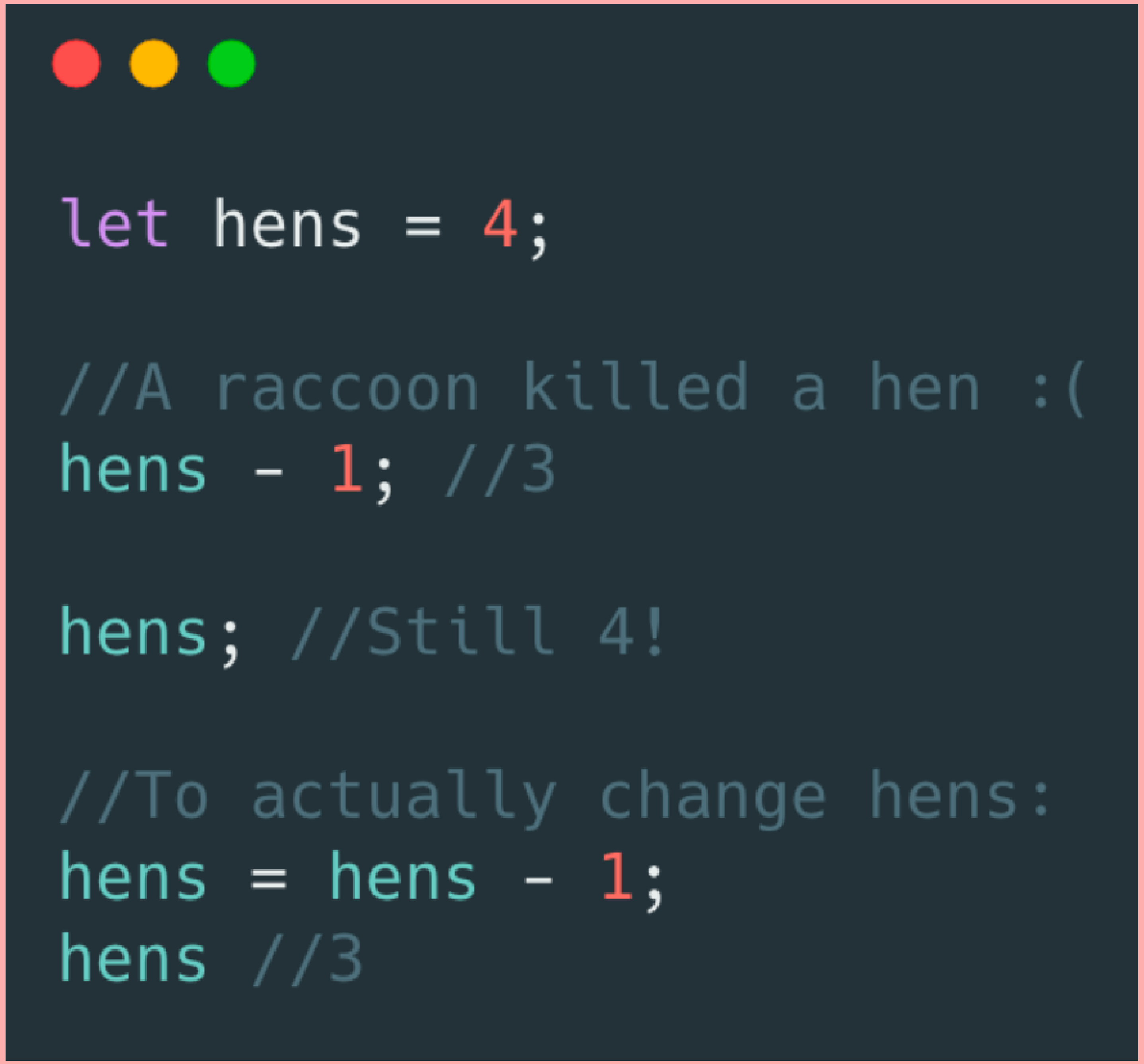
Make me a variable called "year" and give it the value of 1985

RECALL VALUES



```
let hens = 4;

let roosters = 2;

hens + roosters //6
```

RECALL VALUES

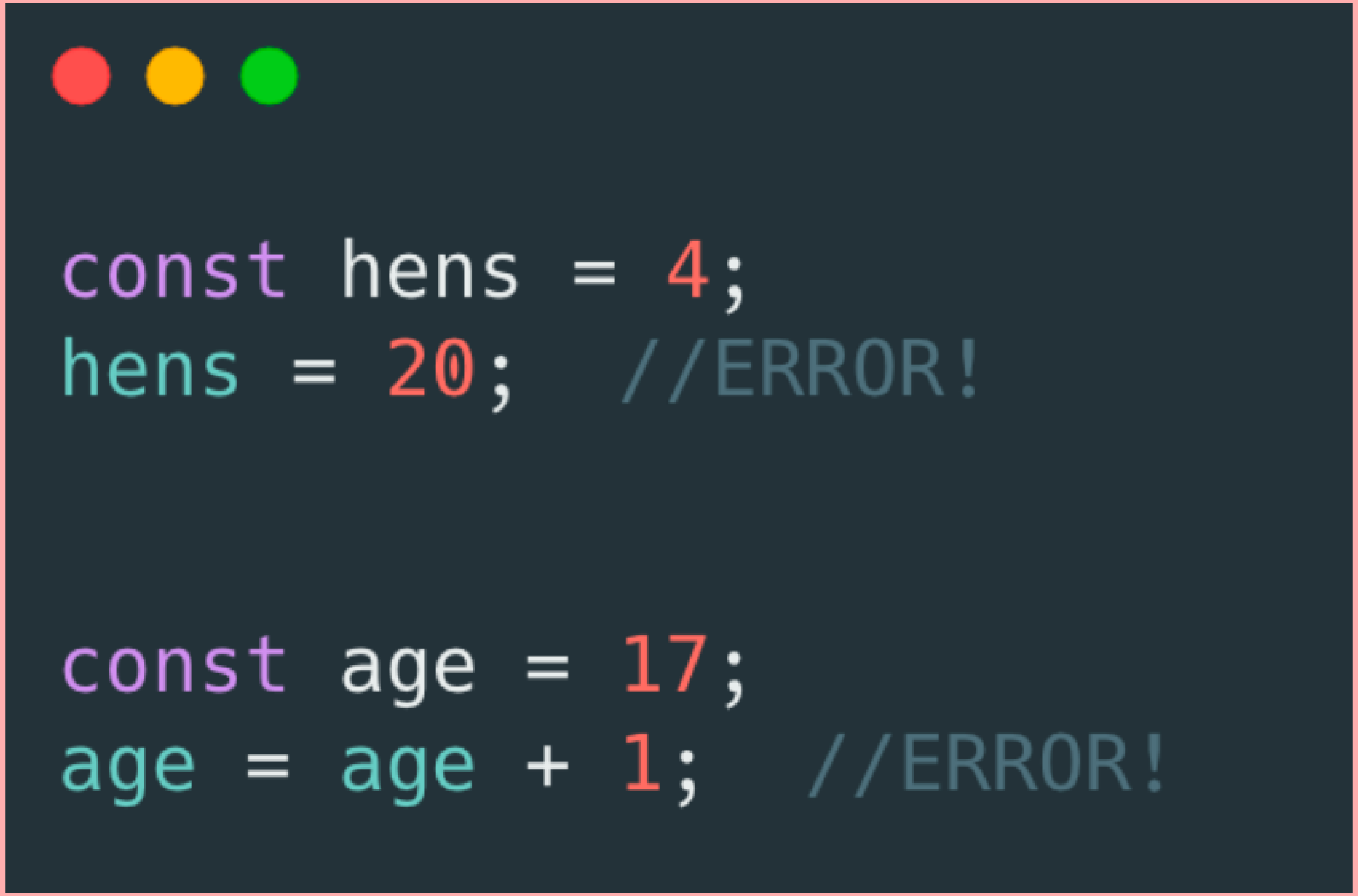


```
let hens = 4;

//A raccoon killed a hen :(
hens - 1; //3

hens; //Still 4!

//To actually change hens:
hens = hens - 1;
hens //3
```

This does not change the value stored in hens

This does!

CONST

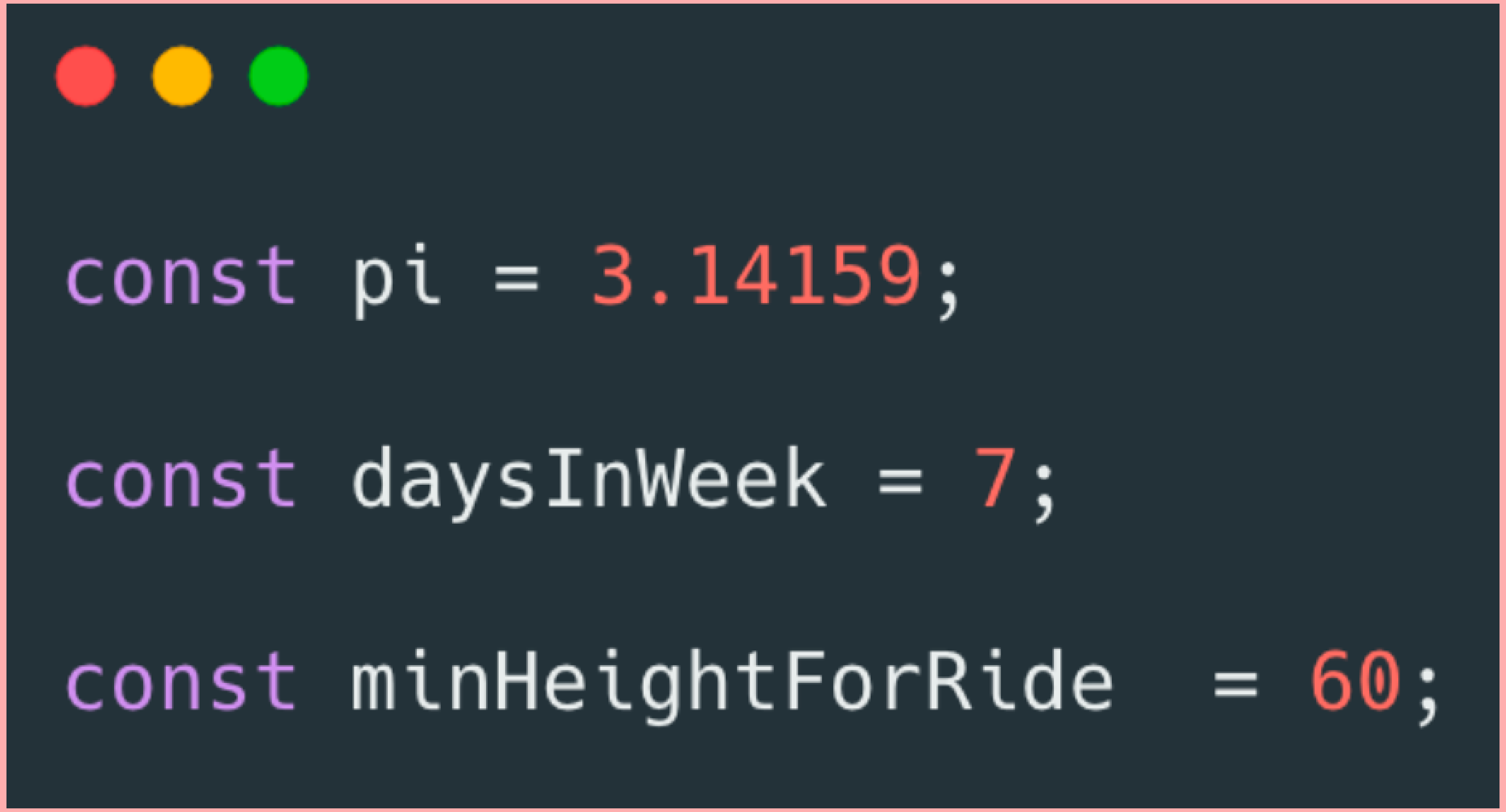


```
const hens = 4;
hens = 20;  //ERROR!


const age = 17;
age = age + 1;  //ERROR!
```

const works just like let, except you CANNOT change the value

NOT ALLOWED!

WHY CONST?

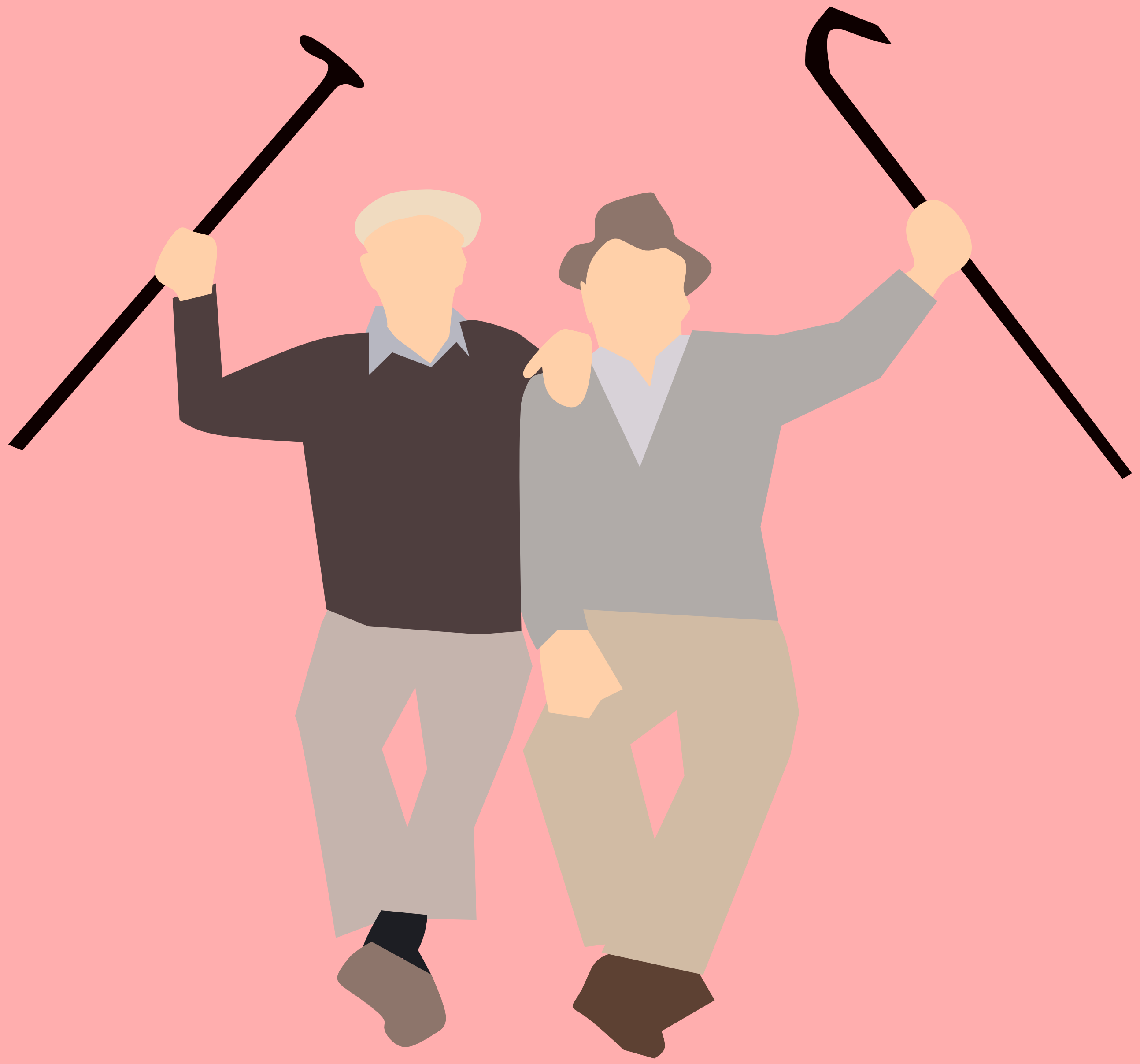


```
const pi = 3.14159;

const daysInWeek = 7;

const minHeightForRide  = 60;
```

In some situations *const* makes sense over *let*.

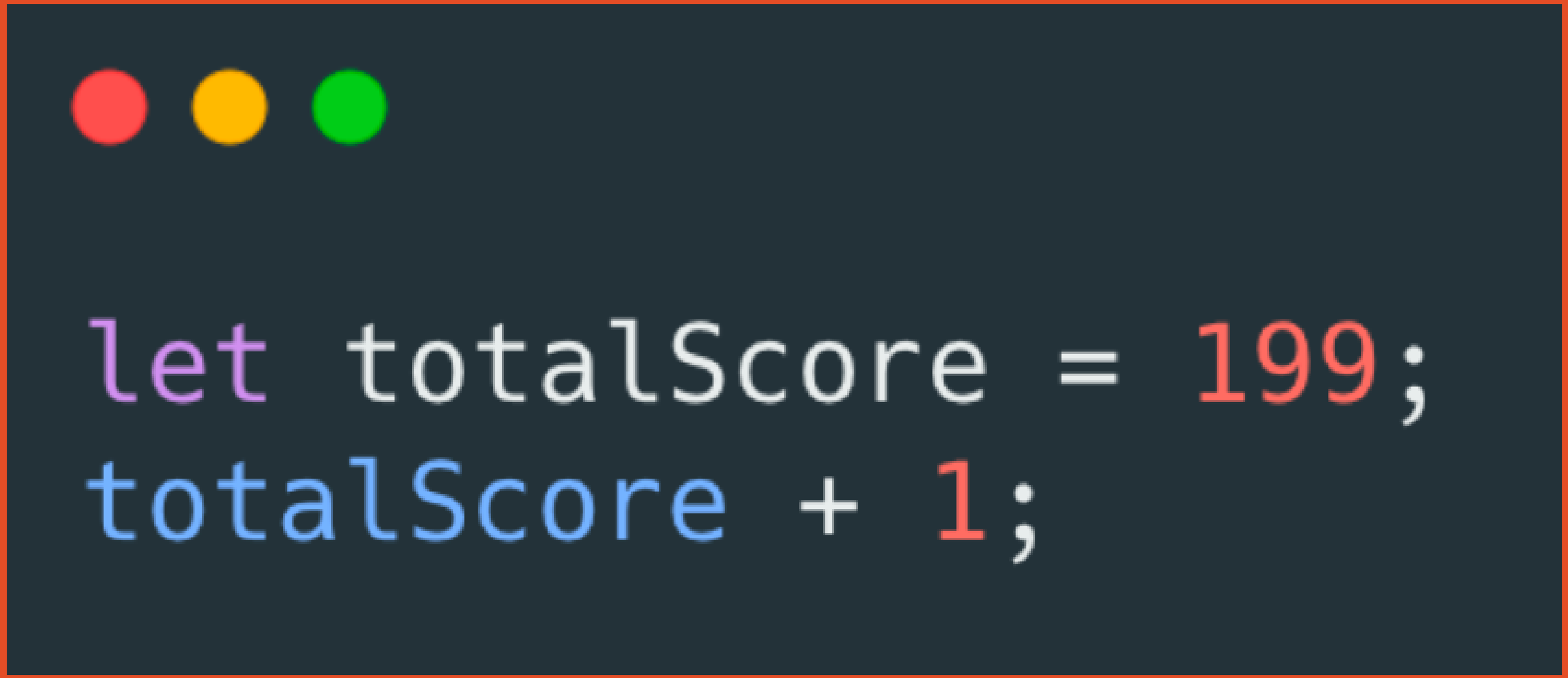
VAR

THE OLD VARIABLE KEYWORD



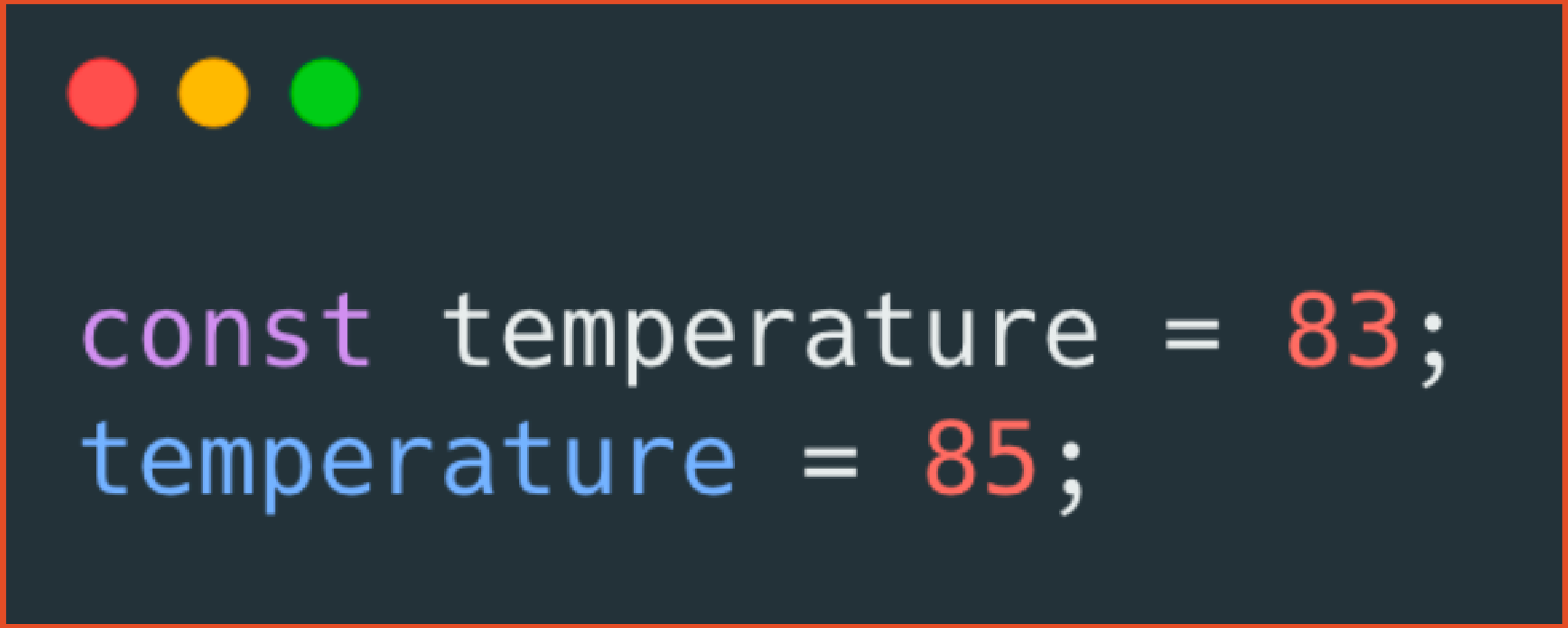
Before let & const, var was the only way of declaring variables. These days, there isnt't really a reason to use it.

WHAT IS THE VALUE OF totalScore?



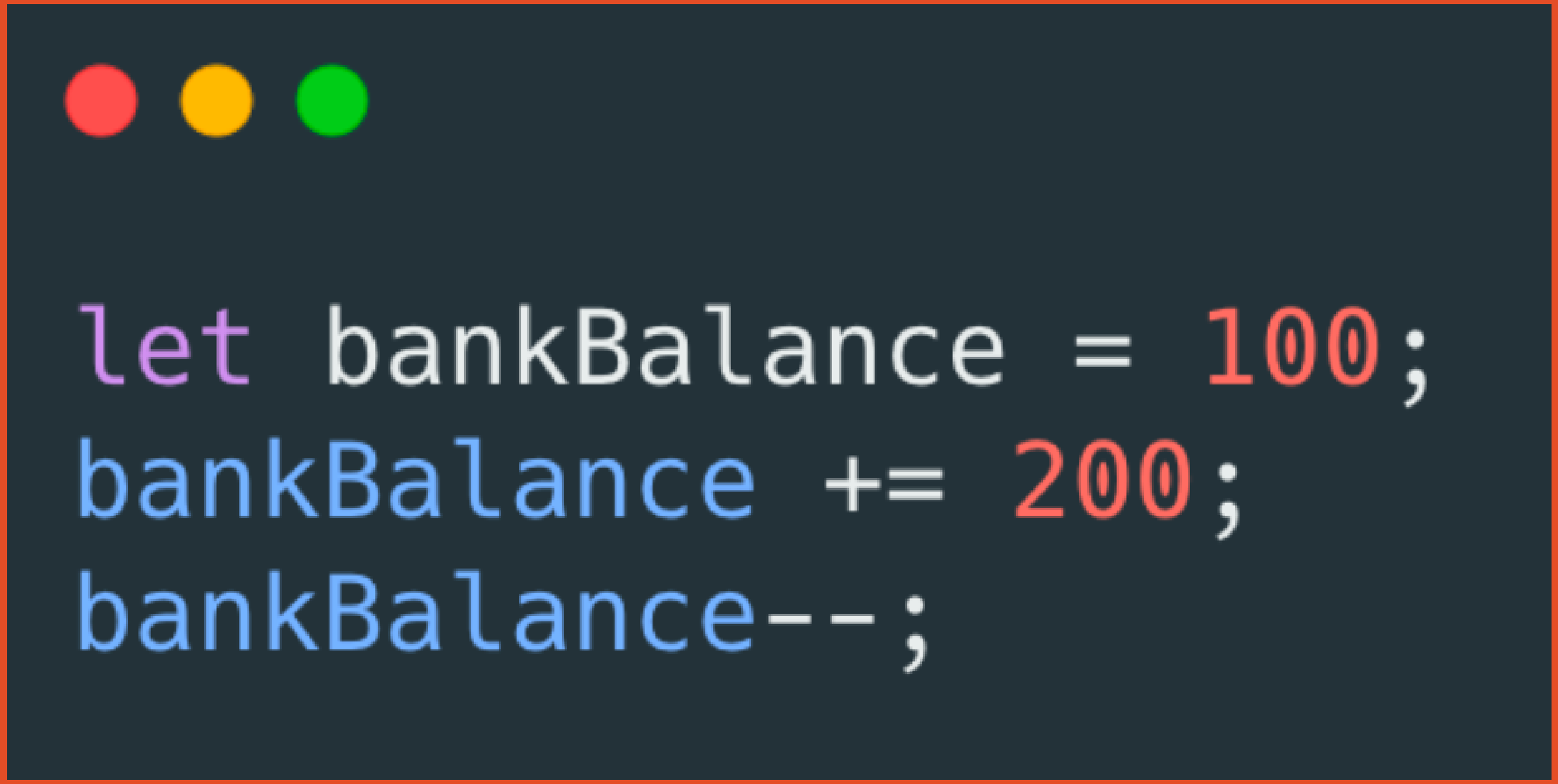
```
let totalScore = 199;
totalScore + 1;
```

WHAT IS THE VALUE OF temperature?



```
const temperature = 83;
temperature = 85;
```

WHAT IS THE VALUE OF bankBalance?

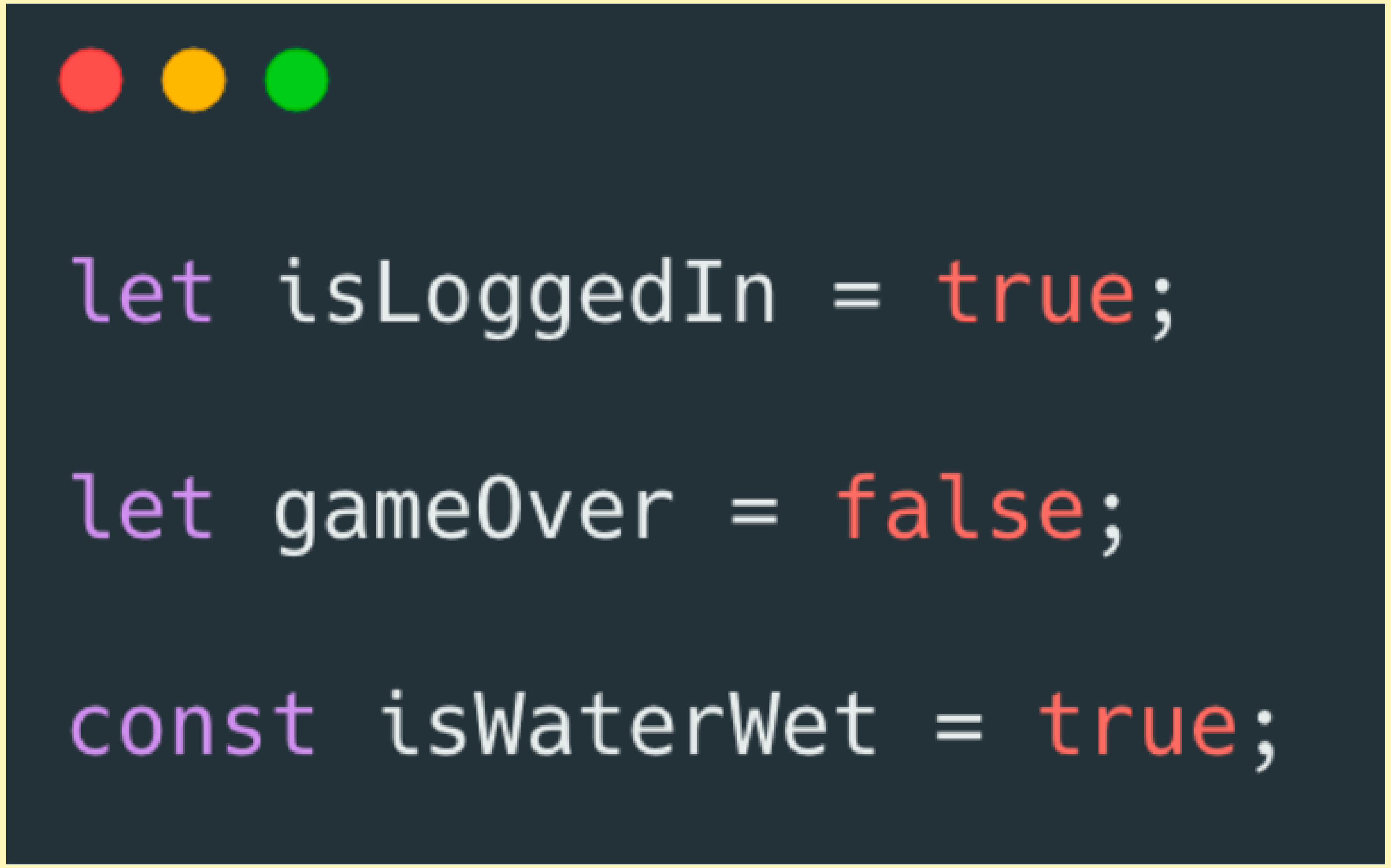


```
let bankBalance = 100;
bankBalance += 200;
bankBalance--;
```

BOOLEANS

OR

BOOLEANS

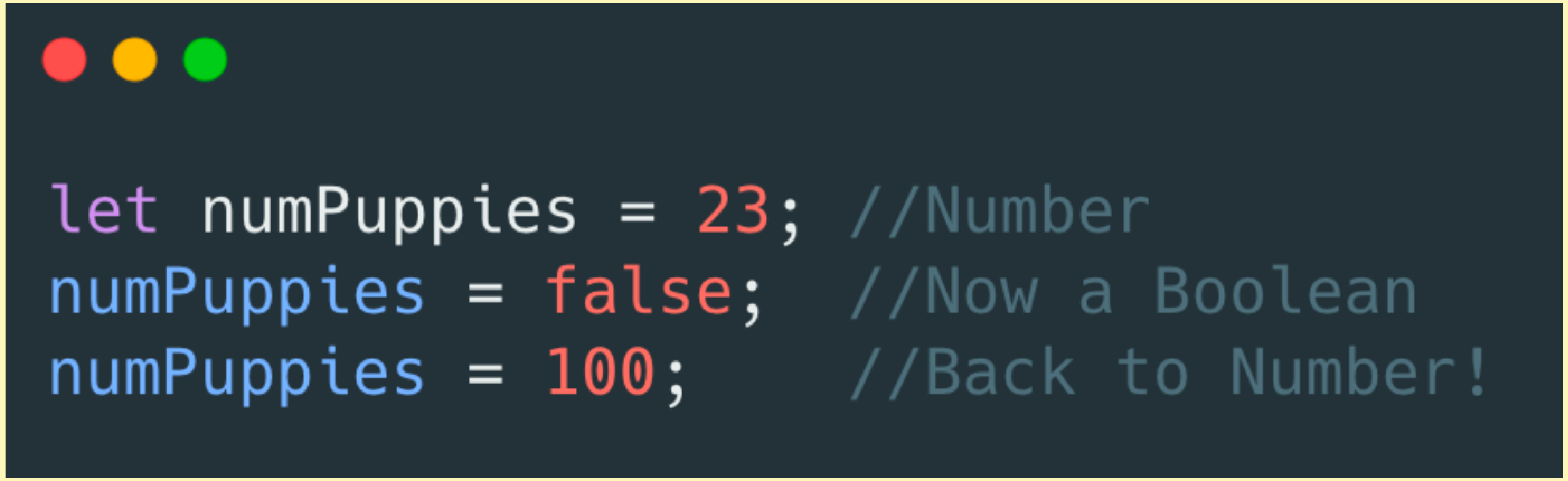


```
let isLoggedIn = true;

let gameOver = false;

const isWaterWet = true;
```

TRUE or FALSE

Booleans are very simple. You have two possible options: true or false. That's it!

VARIABLES CAN CHANGE TYPES



```
let numPuppies = 23; //Number
numPuppies = false;  //Now a Boolean
numPuppies = 100;    //Back to Number!
```

It does not really make sense to change from a number to a boolean here, but we can!