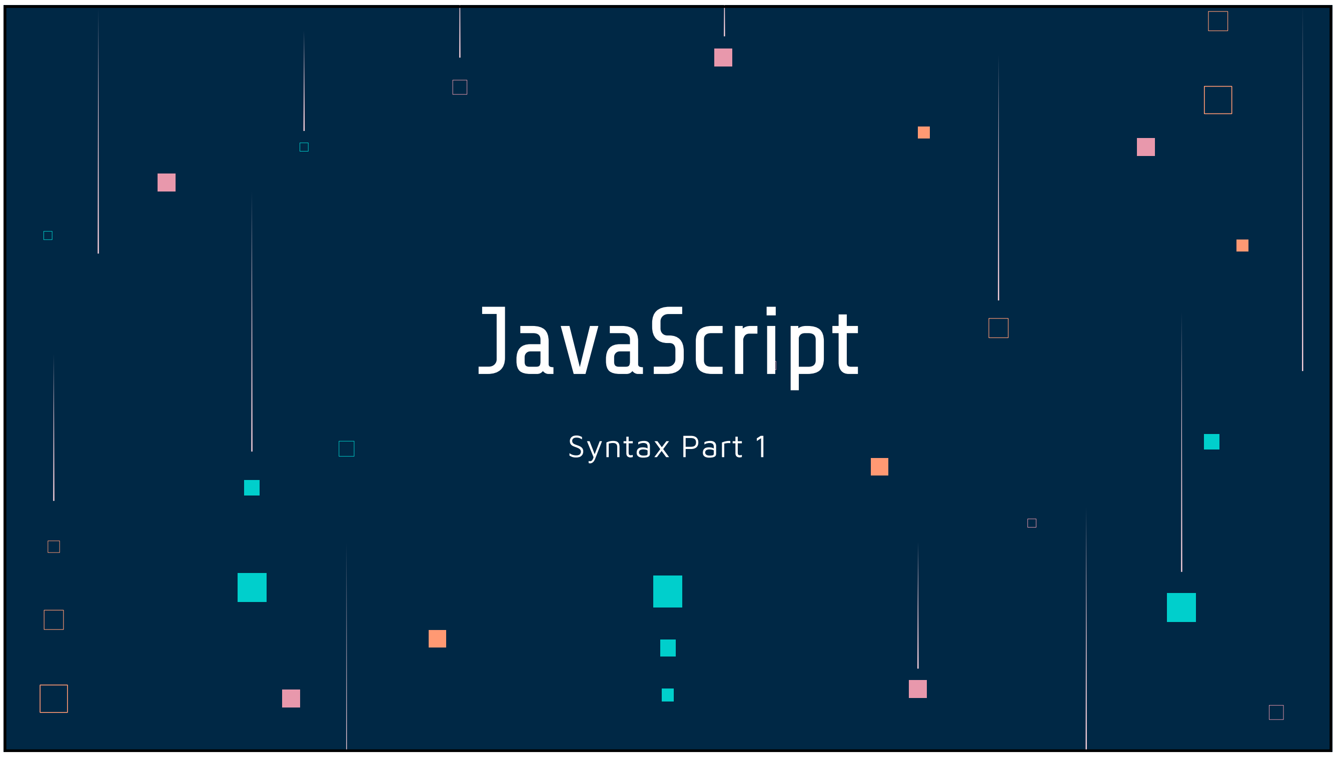
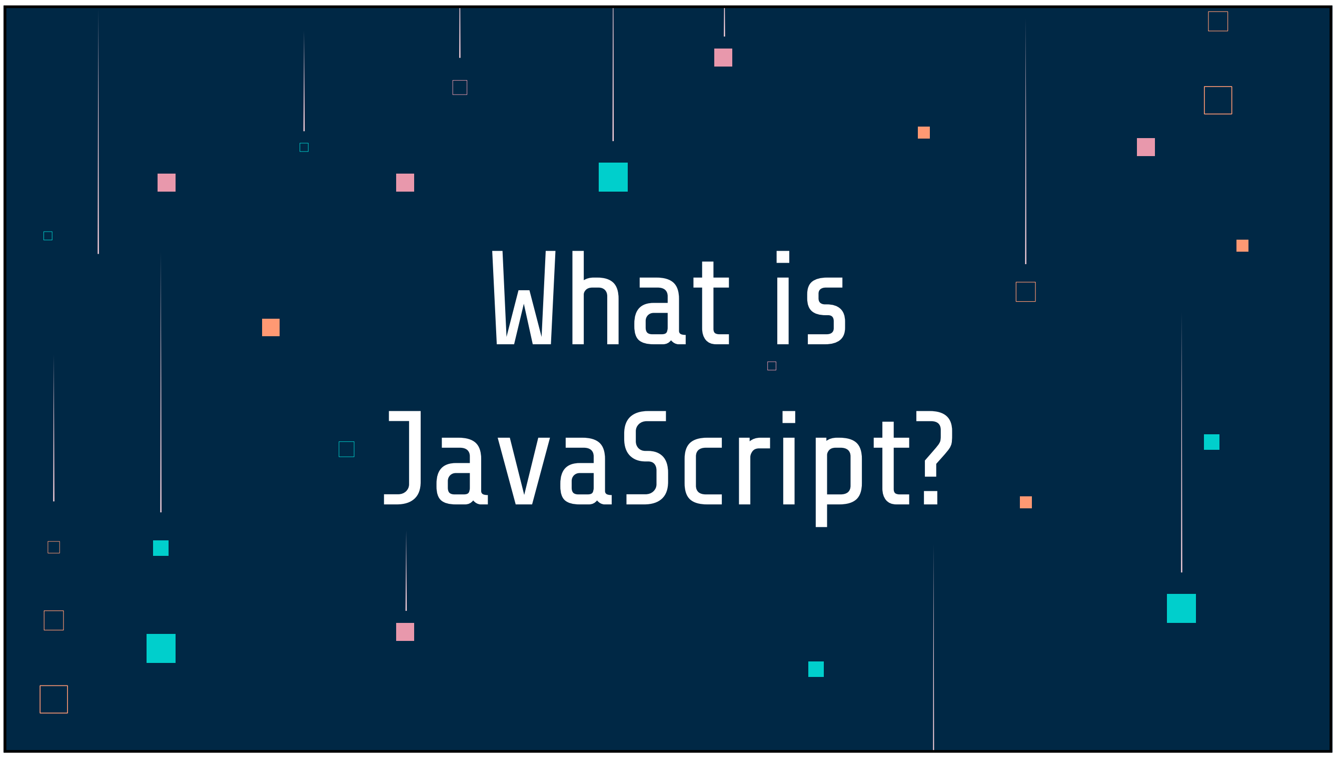


JavaScript

Syntax Part 1

1

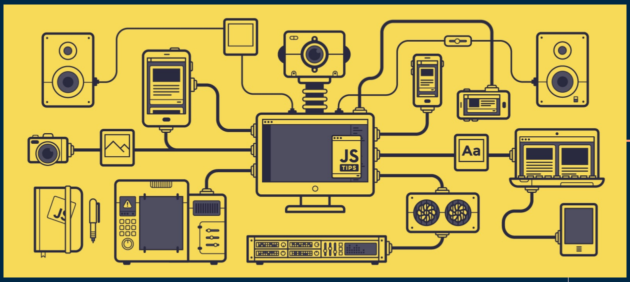


What is JavaScript?

3

JavaScript (JS) is known as the language of most modern web browsers. It is a powerful, flexible, and fast **programming language**.

JS powers the dynamic behaviour on most websites.



4

Is Java and JavaScript the same thing?

Java is to JavaScript as Ham is to Hamster!

Java is a **compiling language**, whereas JS is an **interpreted scripting language**.

Java is used all server-side, whereas JS is limited to making web apps more interactive and creative.

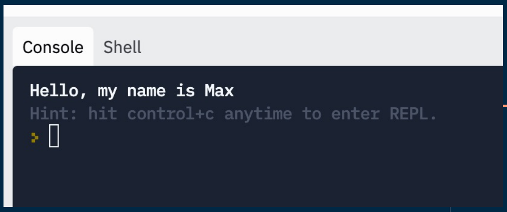
HTML
HTML + CSS
HTML + CSS
+ JAVASCRIPT

5

Console

The **console is a panel** that displays important messages, like errors, for developers. Much of the work the computer does with our code is invisible to us by default. If we want to see things appear on our screen, we can print, or *log*, to our console directly.

In JavaScript, the console keyword refers to an object, a collection of data and actions, that we can use in our code. Keywords are words that are built into the JavaScript language, so the computer will recognise them and treats them specially.



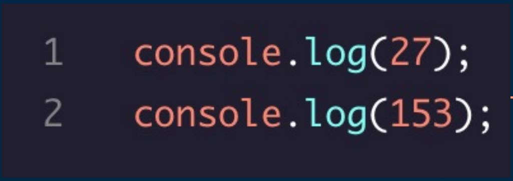
```
Hello, my name is Max
Hint: hit control+c anytime to enter REPL.
›
```

6

Console

One action, or method, that is built into the console object is the **.log()** method. When we write **console.log()** what we put inside the brackets will get printed, or logged, to the console.

1. Use the console.log to log your age to the console.
2. On the next line, write another console.log to print out a different number representing the number of weeks you've been programming.



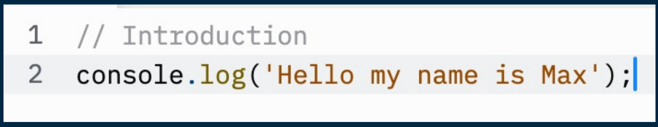
```
console.log(27);
console.log(153);
```

7

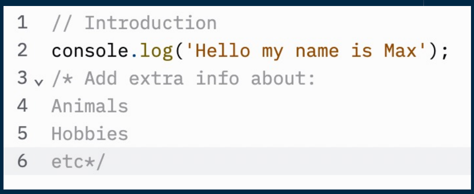
Comments

As we write JavaScript, we can write comments in our code that the computer will ignore as our program runs. These comments exist just for human readers.

Comments can explain what the code is doing, leave instructions for developers using the code, or add any other useful annotations.



```
// Introduction
console.log('Hello my name is Max');
```



```
// Introduction
console.log('Hello my name is Max');
/* Add extra info about:
Animals
Hobbies
etc*/
```

8

Data Types

Data types are the **classifications** we give to the different kinds of data that we use in programming. In JavaScript, there are seven fundamental data types:

Number: Any number, including numbers with decimals: 4, 8, 1516, 23.42.

String: Any grouping of characters on your keyboard (letters, numbers, spaces, symbols, etc.) surrounded by single quotes: ' ... ' or double quotes " ... ". Though we prefer single quotes. Some people like to think of string as a fancy word for text.

Boolean: This data type only has two possible values— either true or false (without quotes). It's helpful to think of booleans as on and off switches or as the answers to a "yes" or "no" question.

9

Data Types

Null: This data type represents the intentional absence of a value, and is represented by the keyword null (without quotes).

Undefined: This data type is denoted by the keyword undefined (without quotes). It also represents the absence of a value though it has a different use than null.

Symbol: A newer feature to the language, symbols are unique identifiers, useful in more complex coding. No need to worry about these for now.
Object: Collections of related data.

10

Data Types

The first 6 of those types are considered *primitive data types*. They are the most basic data types in the language. *Objects* are more complex, and you'll learn much more about them as you progress through JavaScript.

As you learn more about objects, you'll be able to create complex collections of data.

But before we do that, let's get comfortable with strings and numbers!

11

Strings and Numbers

In the example, we first printed a string. Our string isn’t just a single word; it includes both capital and lowercase letters, spaces, and punctuation.

Next, we printed the number 40, notice we did not use quotes.

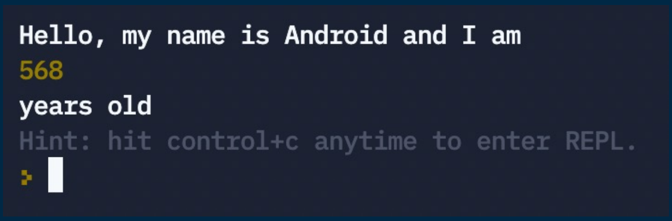
```
console.log('Location of TechTalent Academy: 57
Colvile Row, Birmingham');
console.log(40);
```

12

Task 2

On line 1: Print *Hello, my name is Android and I am*
On line 2: Print *568*
On line 3: Print *years old*

Console.log



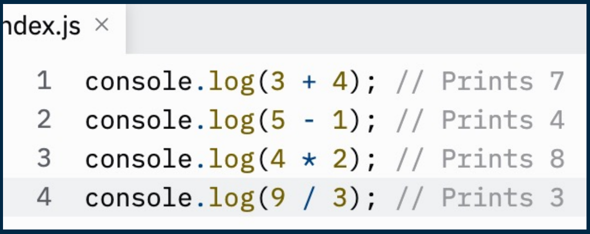
```
Hello, my name is Android and I am
568
years old
Hint: hit control+c anytime to enter REPL.
›
```

13

Arithmetic Operators

An operator is a character that performs a task in our code. JavaScript has several built-in *arithmetic operators,* that allow us to perform mathematical calculations on numbers. These include the following operators and their corresponding symbols:

Add: +
Subtract: -
Multiply: *
Divide: /
Remainder: %


ndex.js ×

```
console.log(3 + 4); // Prints 7
console.log(5 - 1); // Prints 4
console.log(4 * 2); // Prints 8
console.log(9 / 3); // Prints 3
```

14

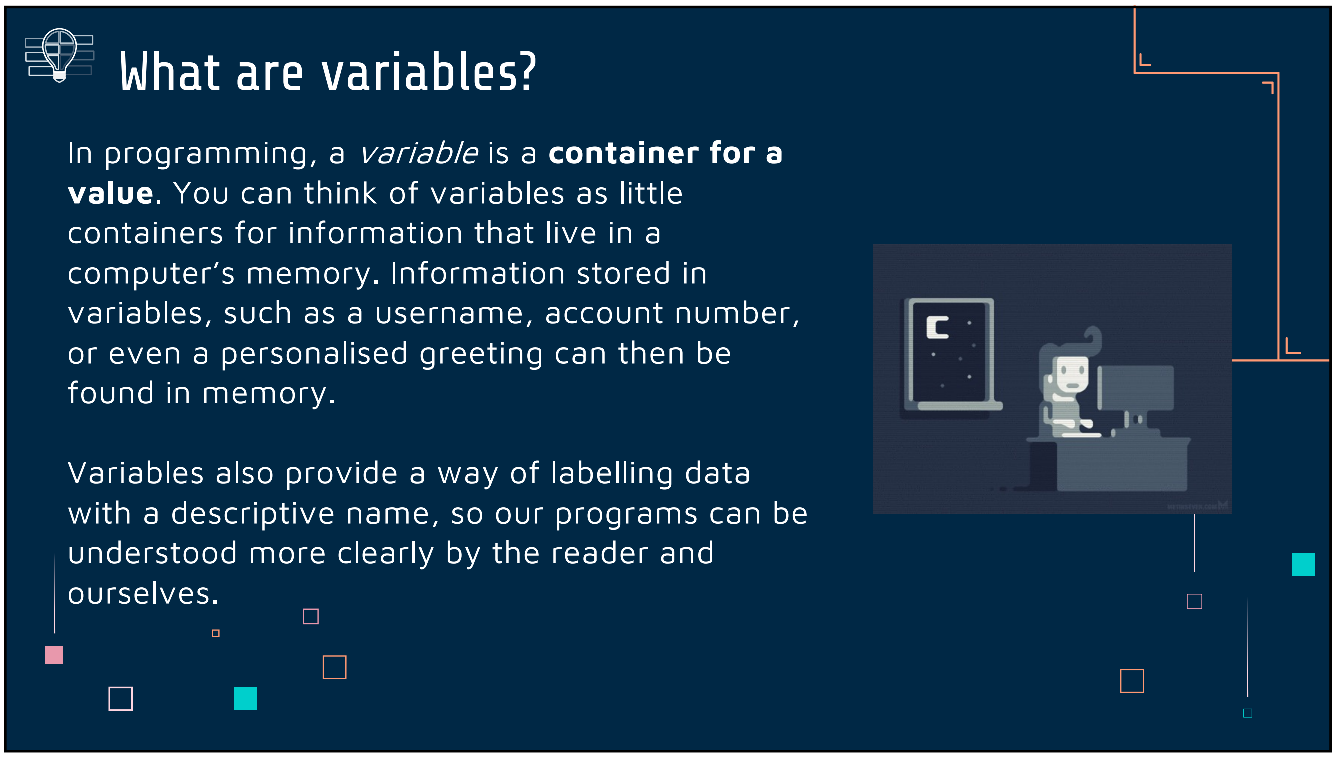
Task 3

1. Inside a console.log(), **add 3.5** to your age. (the age you'll be when we start sending people to live on Mars)
2. On another line (console.log), **take the current year and subtract 1969** (how many years since the 1969 moon landing)
3. On another line (console.log), inside the brackets **divide 65 by 240**.
4. On your final line (console.log) **multiply 0.2708 by 123** (the % of the sun that is made up of helium!)

15

Variables

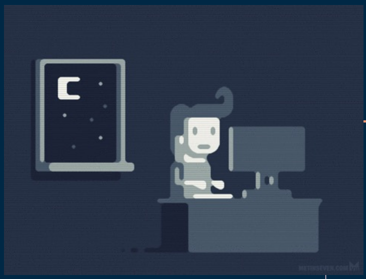
16



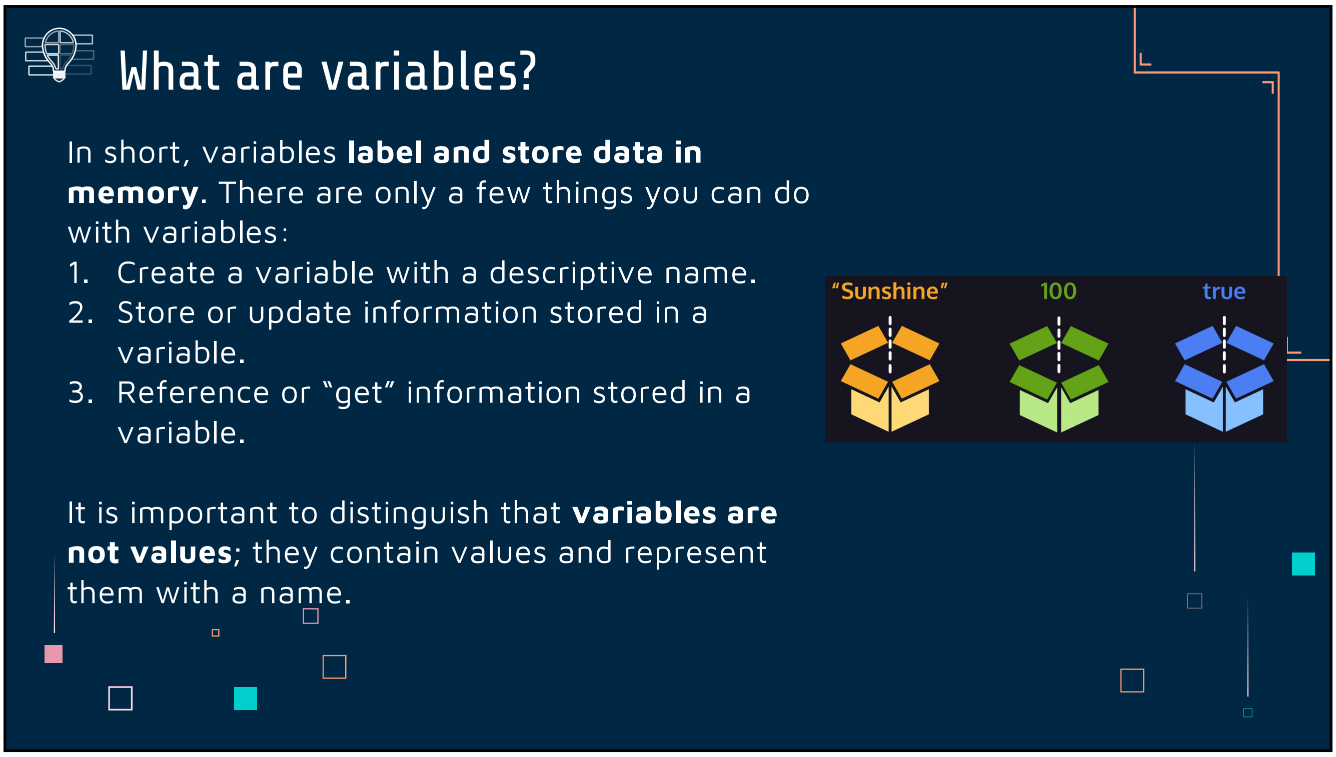
What are variables?

In programming, a *variable* is a **container for a value**. You can think of variables as little containers for information that live in a computer's memory. Information stored in variables, such as a username, account number, or even a personalised greeting can then be found in memory.

Variables also provide a way of labelling data with a descriptive name, so our programs can be understood more clearly by the reader and ourselves.



17



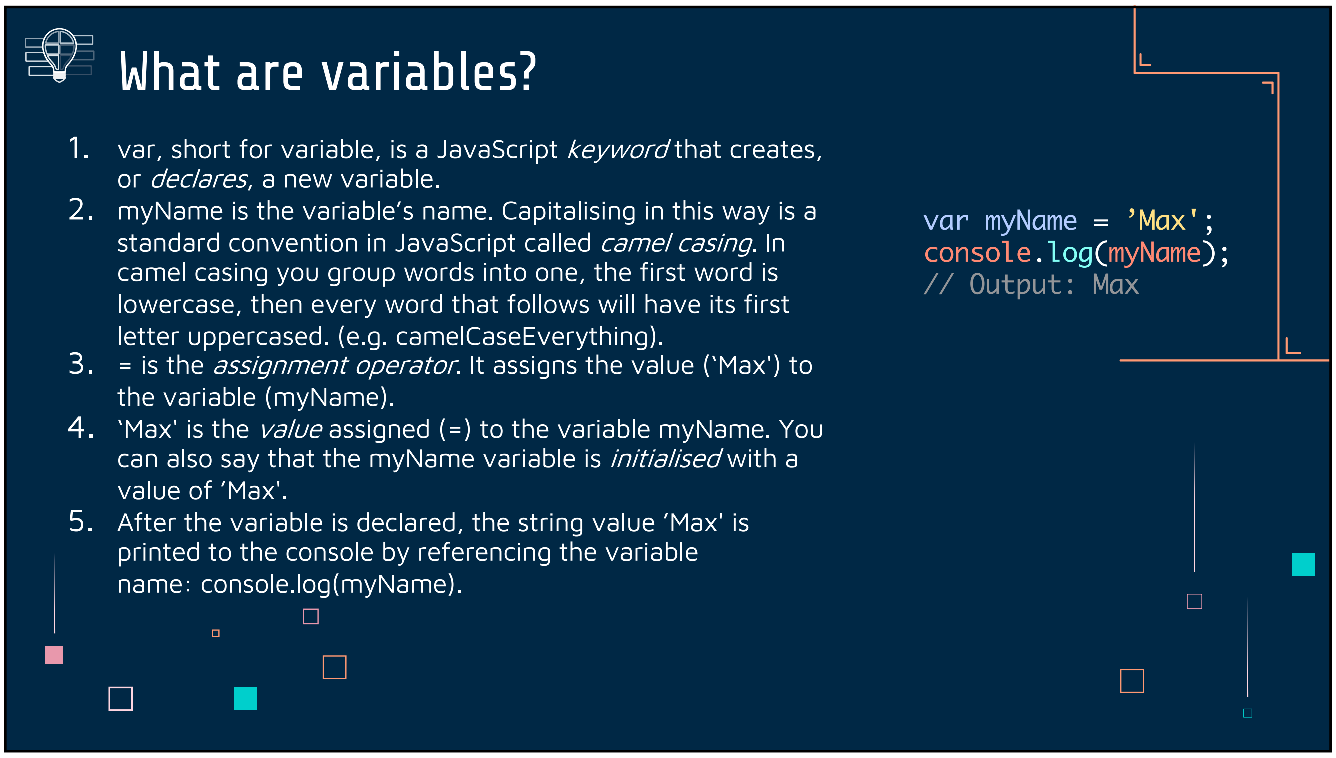
What are variables?

In short, variables **label and store data in memory**. There are only a few things you can do with variables:

1. Create a variable with a descriptive name.
2. Store or update information stored in a variable.
3. Reference or "get" information stored in a variable.

It is important to distinguish that **variables are not values**; they contain values and represent them with a name.

18

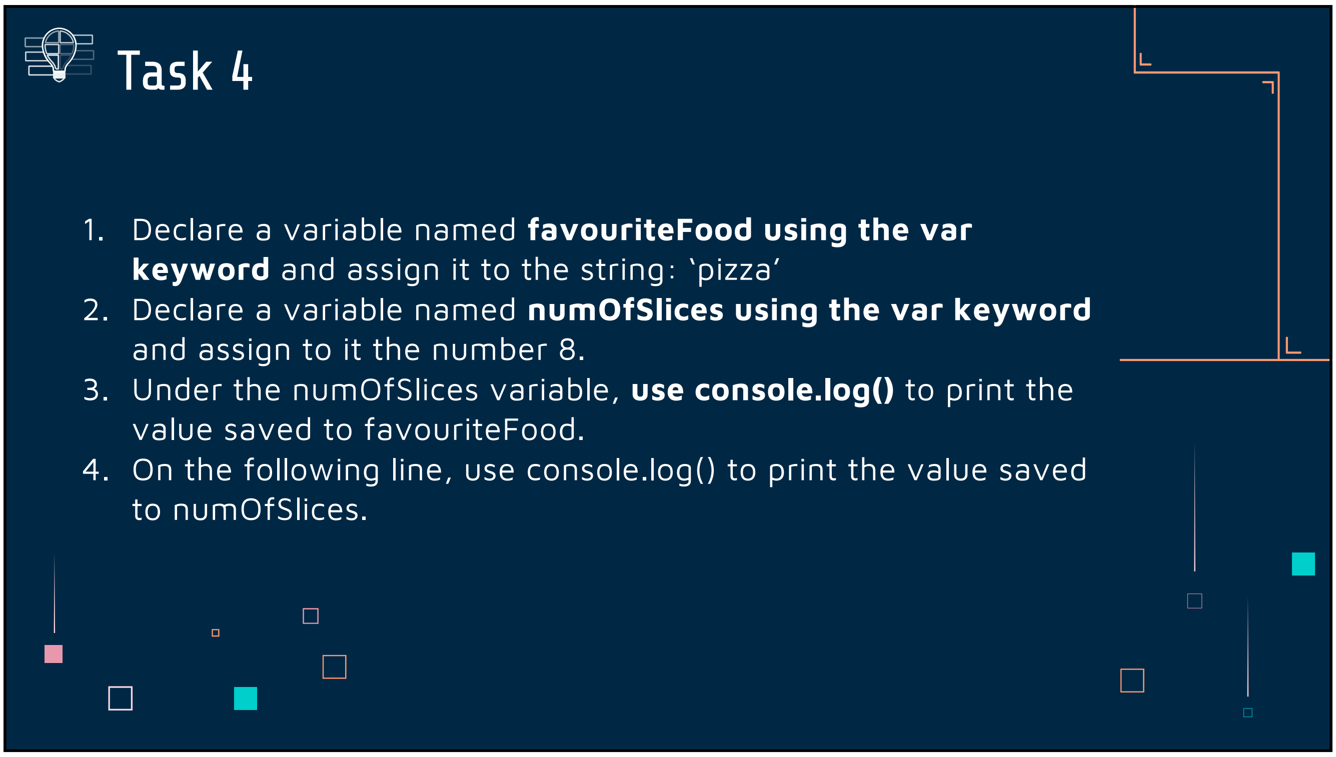


What are variables?

1. var, short for variable, is a JavaScript *keyword* that creates, or *declares*, a new variable.
2. myName is the variable's name. Capitalising in this way is a standard convention in JavaScript called *camel casing*. In camel casing you group words into one, the first word is lowercase, then every word that follows will have its first letter uppercased. (e.g. camelCaseEverything).
3. = is the *assignment operator*. It assigns the value ('Max') to the variable (myName).
4. 'Max' is the *value* assigned (=) to the variable myName. You can also say that the myName variable is *initialised* with a value of 'Max'.
5. After the variable is declared, the string value 'Max' is printed to the console by referencing the variable name: console.log(myName).

```
var myName = 'Max';
console.log(myName);
// Output: Max
```

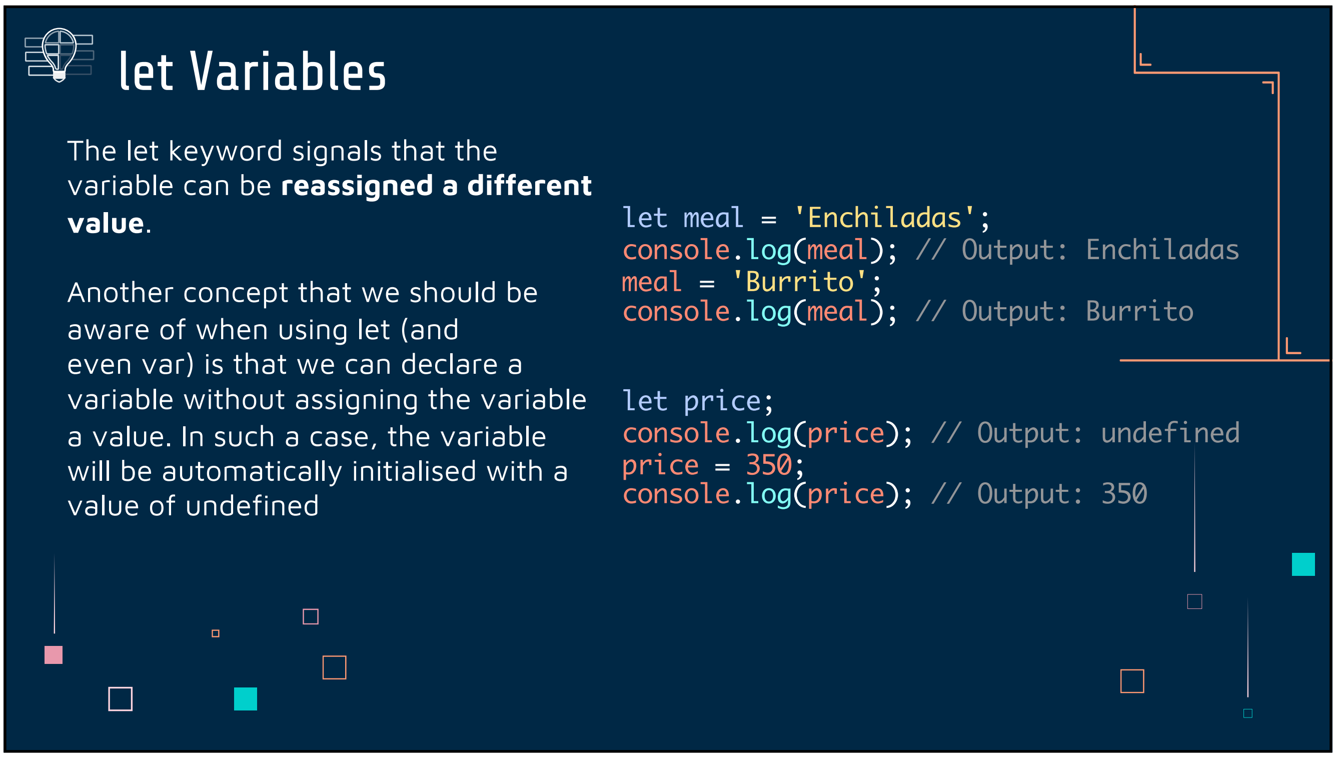
19



Task 4

1. Declare a variable named **favouriteFood using the var keyword** and assign it to the string: ‘pizza’
2. Declare a variable named **numOfSlices using the var keyword** and assign to it the number 8.
3. Under the numOfSlices variable, **use console.log()** to print the value saved to favouriteFood.
4. On the following line, use console.log() to print the value saved to numOfSlices.

20



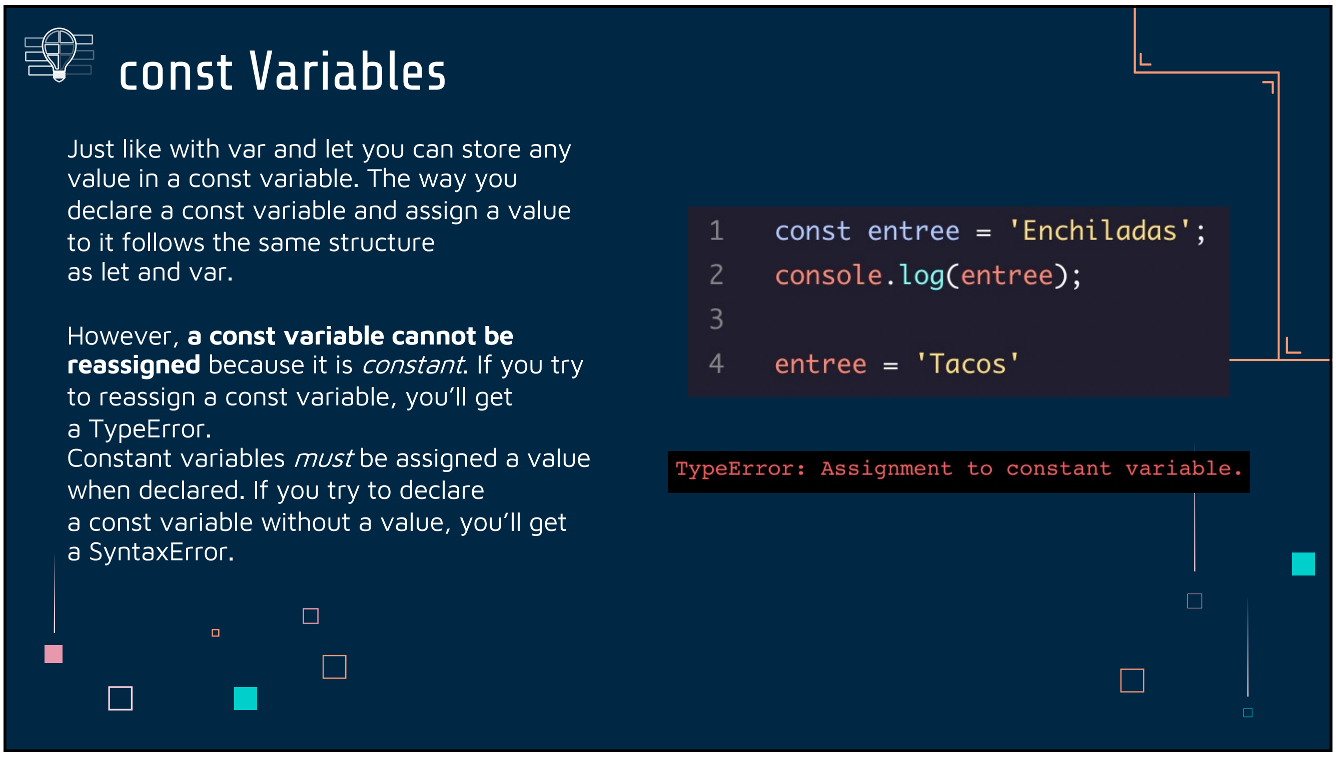
let Variables

The let keyword signals that the variable can be **reassigned a different value**.

Another concept that we should be aware of when using let (and even var) is that we can declare a variable without assigning the variable a value. In such a case, the variable will be automatically initialised with a value of undefined

```
let meal = 'Enchiladas';
console.log(meal); // Output: Enchiladas
meal = 'Burrito';
console.log(meal); // Output: Burrito

let price;
console.log(price); // Output: undefined
price = 350;
console.log(price); // Output: 350
```

21



const Variables

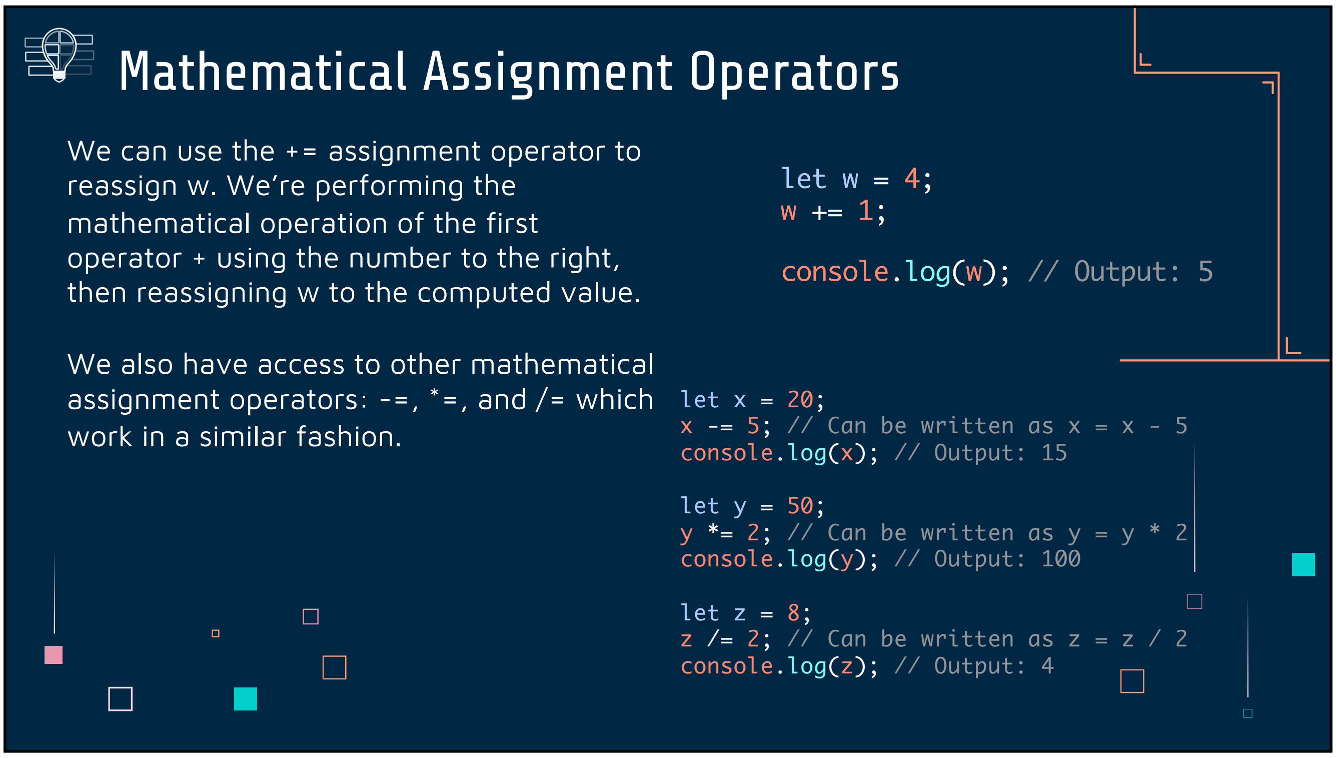
Just like with var and let you can store any value in a const variable. The way you declare a const variable and assign a value to it follows the same structure as let and var.

However, **a const variable cannot be reassigned** because it is *constant*. If you try to reassign a const variable, you'll get a TypeError.
Constant variables *must* be assigned a value when declared. If you try to declare a const variable without a value, you'll get a SyntaxError.

```
const entree = 'Enchiladas';
console.log(entree);

entree = 'Tacos'
```

```
TypeError: Assignment to constant variable.
```

22



Mathematical Assignment Operators

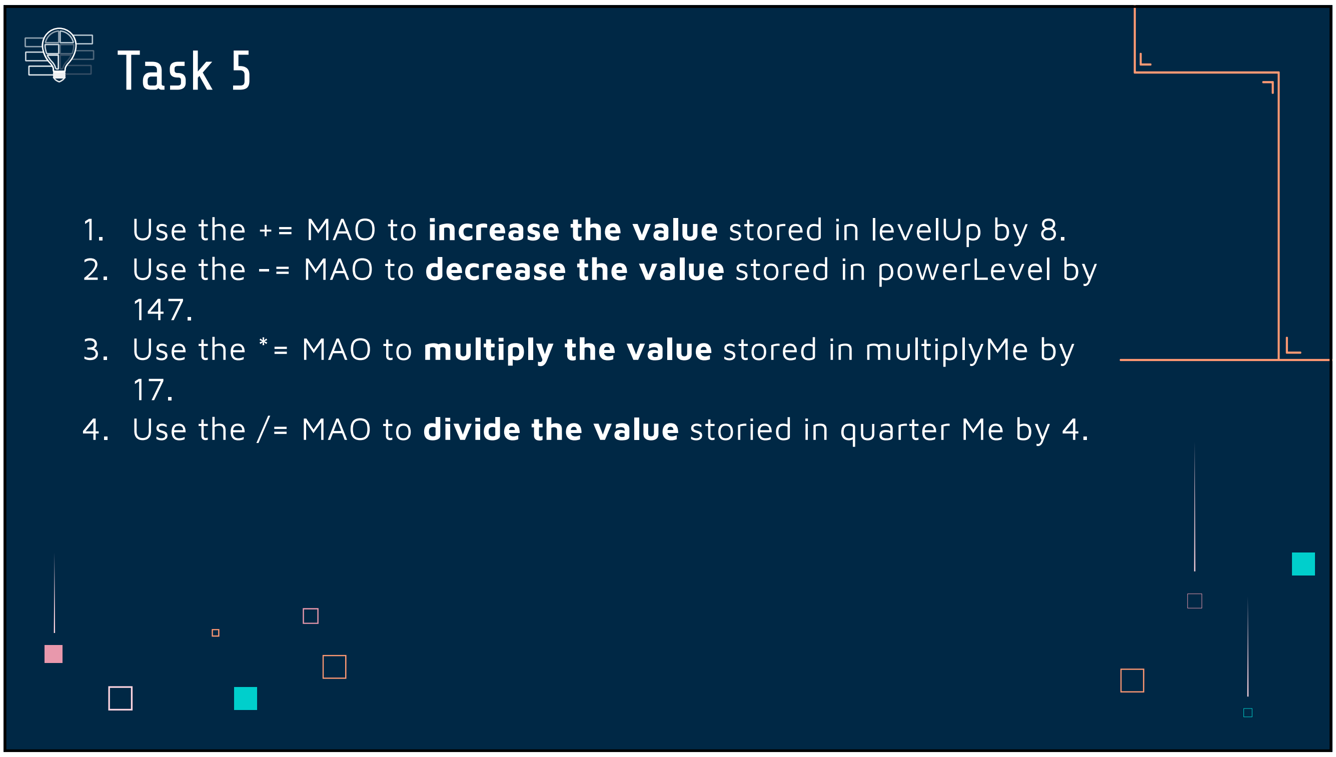
We can use the += assignment operator to reassign w. We're performing the mathematical operation of the first operator + using the number to the right, then reassigning w to the computed value.

We also have access to other mathematical assignment operators: -=, *=, and /= which work in a similar fashion.

```
let w = 4;
w += 1;

console.log(w); // Output: 5
```

```
let x = 20;
x -= 5; // Can be written as x = x - 5
console.log(x); // Output: 15

let y = 50;
y *= 2; // Can be written as y = y * 2
console.log(y); // Output: 100

let z = 8;
z /= 2; // Can be written as z = z / 2
console.log(z); // Output: 4
```

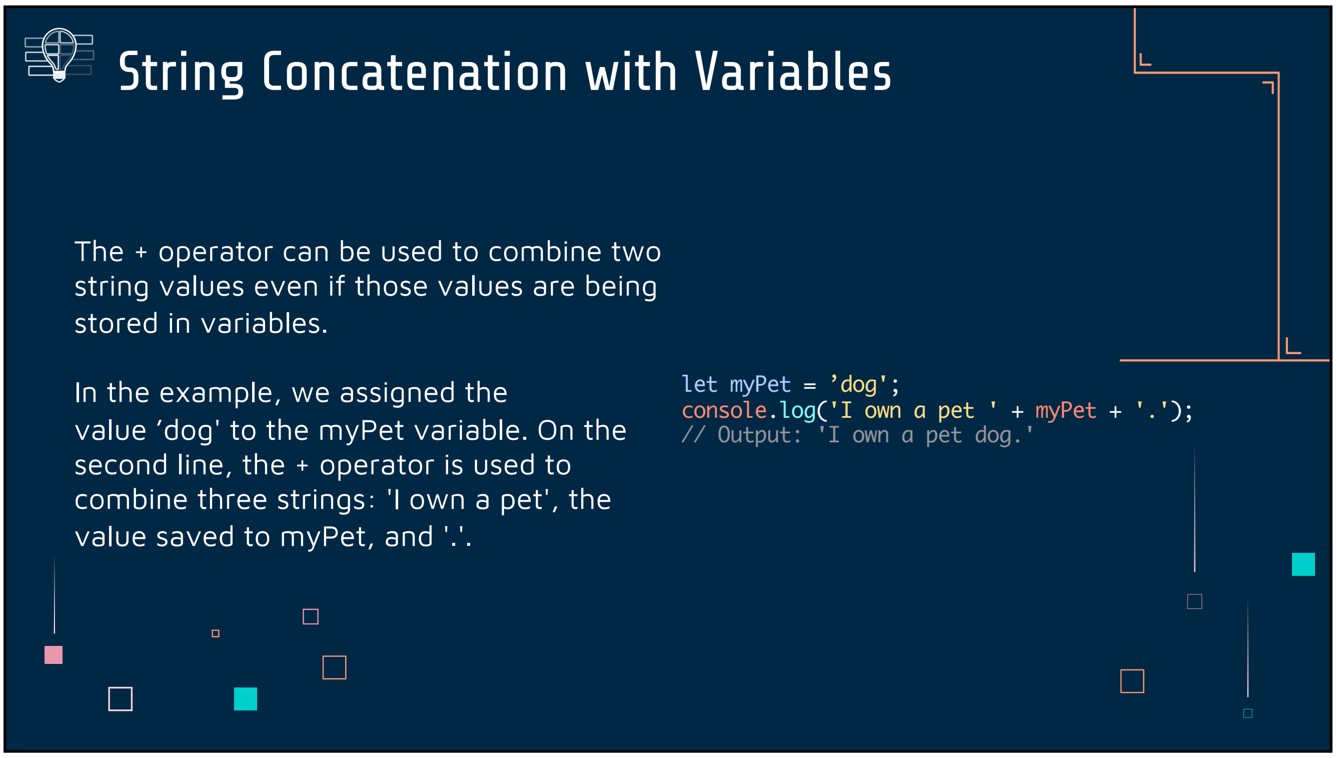
23



Task 5

1. Use the += MAO to **increase the value** stored in levelUp by 8.
2. Use the -= MAO to **decrease the value** stored in powerLevel by 147.
3. Use the *= MAO to **multiply the value** stored in multiplyMe by 17.
4. Use the /= MAO to **divide the value** storied in quarter Me by 4.

24



String Concatenation with Variables

The + operator can be used to combine two string values even if those values are being stored in variables.

In the example, we assigned the value ‘dog' to the myPet variable. On the second line, the + operator is used to combine three strings: 'I own a pet', the value saved to myPet, and '.'.

```
let myPet = 'dog';
console.log('I own a pet ' + myPet + '.');
// Output: 'I own a pet dog.'
```

25

String Interpolation

In JS we can insert (interpolate) variables into strings using template literals. A template literal is:

- Wrapped by backticks
- Contains a placeholder (${})

One of the biggest benefits to using template literals is the **readability of the code.**

```
const myName = 'Max';
let myPet = 'dog';
let myPetName = 'Flynn';
const myPetGender = 'his';
console.log(`My name is ${myName}. I have a ${myPet} and $
{myPetGender} name is ${myPetName}.`);
```

```
My name is Max. I have a dog and his name is Flynn.
```

26

Task 6

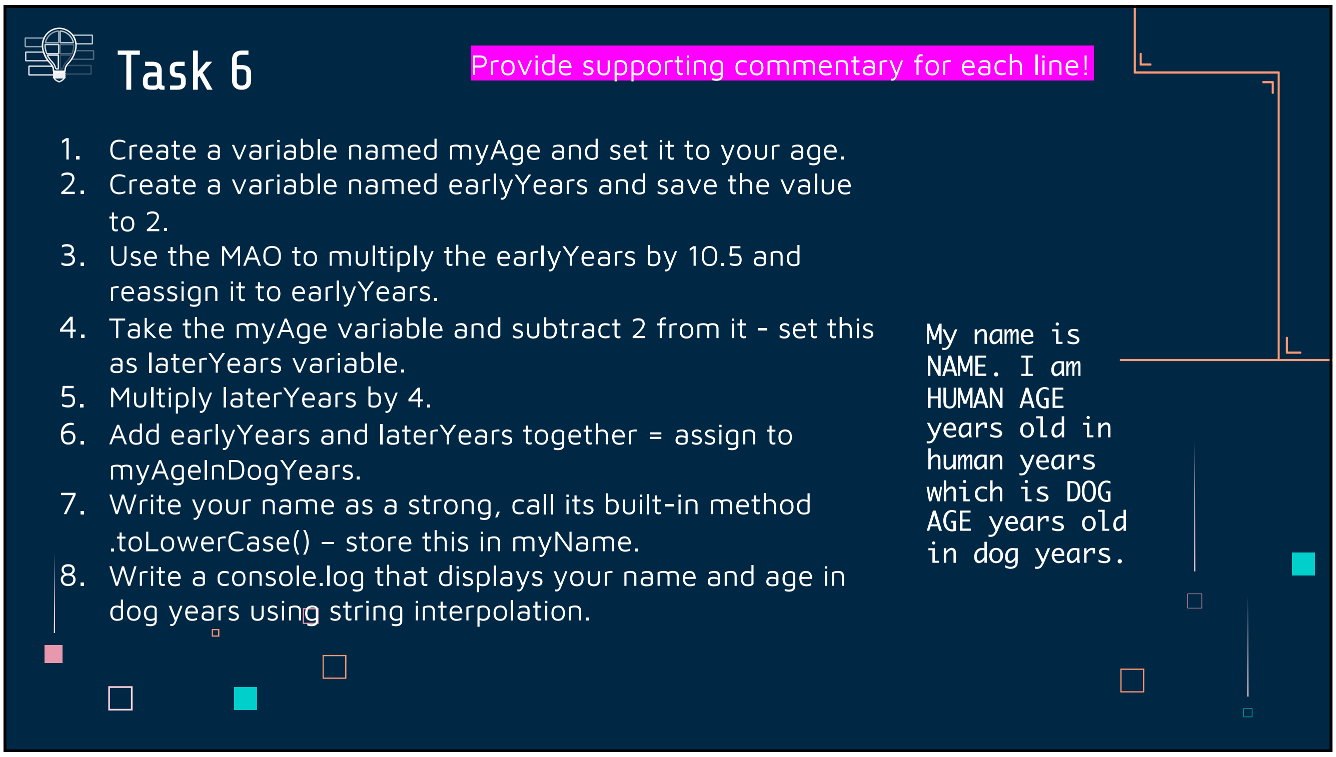
Dogs mature at a faster rate than human beings. We often say a dog's age can be calculated in "dog years" to account for their growth compared to a human of the same age. In some ways we could say, time moves quickly for dogs — 8 years in a human's life equates to 45 years in a dog's life. How old would you be if you were a dog?

Here's how you convert your age from "human years" to "dog years":

- The first two years of a dog's life count as 10.5 dog years each.
- Each year following equates to 4 dog years.

With your knowledge of math operators and variables, use JavaScript to convert your human age into dog years.

27



Task 6

Provide supporting commentary for each line!

1. Create a variable named myAge and set it to your age.
2. Create a variable named earlyYears and save the value to 2.
3. Use the MAO to multiply the earlyYears by 10.5 and reassign it to earlyYears.
4. Take the myAge variable and subtract 2 from it - set this as laterYears variable.
5. Multiply laterYears by 4.
6. Add earlyYears and laterYears together = assign to myAgeInDogYears.
7. Write your name as a strong, call its built-in method .toLowerCase() – store this in myName.
8. Write a console.log that displays your name and age in dog years using string interpolation.

```
My name is
NAME. I am
HUMAN AGE
years old in
human years
which is DOG
AGE years old
in dog years.
```

28