# the Master Course

{CODENATION}

# JAVASCRIPT FUNDAMENTALS

## Variables

{CODENATION}

## Learning Objectives

To understand and use variables and operators to store values and manipulate them

To use camelCase when naming variables

To understand how to access data in variables

## First Things First!

Display the **8th character** of this sentence in upper case on the console.

**All Around the World**

**Hint**: Look at charAt()

{CODENATION}

```
console.log("All Around the
world".charAt(7).toUpperCase());
```

JS

# Introducing

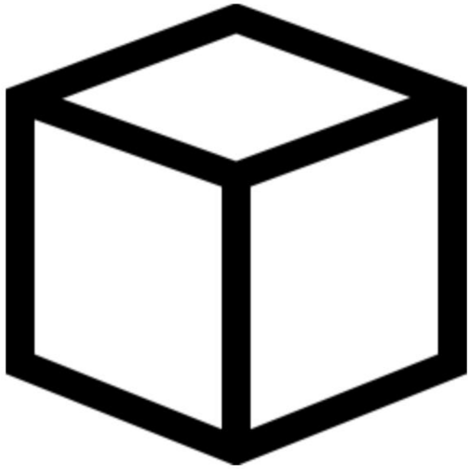
... Variables!

{CODENATION}

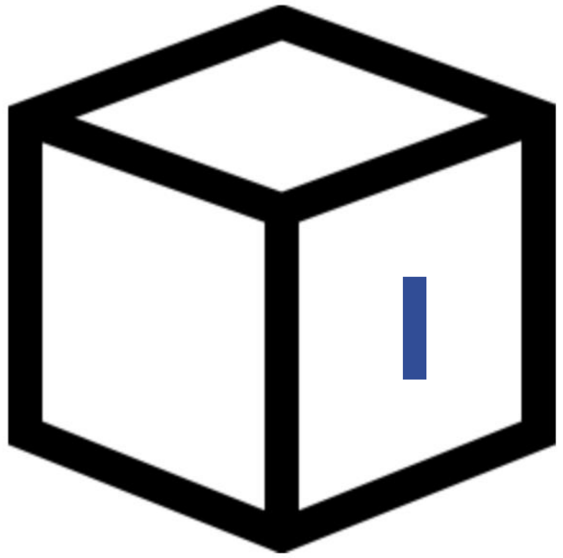
JS

## They're like boxes

... not very **technical** is it?



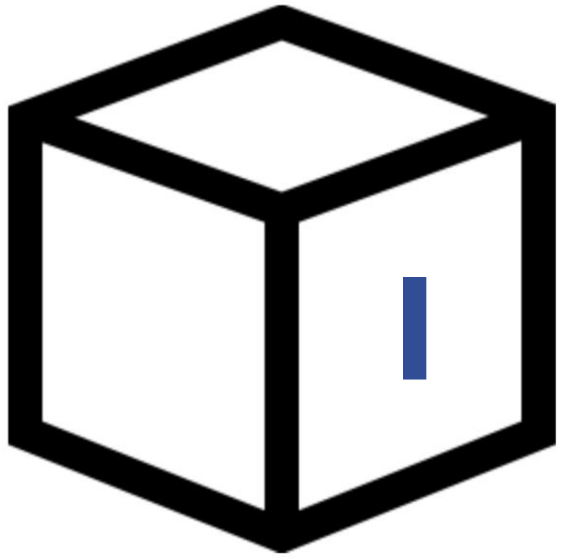
{CODENATION}



We **store items** in boxes to **retrieve later**.

**Different items** can be stored in the box at **different times**.

## So variables...



We **store items** in boxes to retrieve later.

In code we **give variables names** so we can **access things inside them**!

## Imagine a Cash Machine

This is <u>hard</u> coded

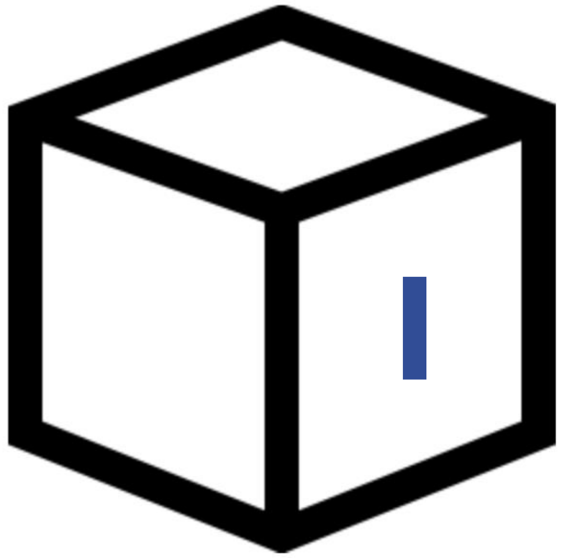
**WITHDRAW 10_POUNDS FROM 82929201**

Should be

This is <u>dynamic</u>

**WITHDRAW AMOUNT FROM ACCNUM**

… how can we make sure we can **reuse code?**



1. Allow us to **store data inside them**
2. Access them **via a name**
3. **Place new data in them** whenever we want

## Javascript is a

… **dynamically typed** language.

We **don't need to tell it** the type of data we are storing in our variable. **It just knows!**

{CODENATION}

## How can we declare a variable

**let**

...is used for declaring a value that **CAN** be changed

**const**

...is used for declaring a value that **CANNOT** be changed. Const = Constant

**var**

...is used for declaring a value that **CAN** be changed. However, it is considered a legacy command now.

## let

```
let i = 10;
```

## const

```
const i = 10;
```

## var

```
var i = 10;
```

JS

let & const = 🤩

var = 👴 ✕

{CODENATION}

JS

Let's look again at...

## Data Types

{CODENATION}

JS

## Strings

… for representing **text**

## Numbers

… for representing **numbers** (decimals & integers)

## Boolean

… for **true** or **false**

## Null

… for **nothing**

## Undefined

… for when a data type **isn't determined**

## Symbol

… this data type is used as the key for an object property when the property is intended to be private.

{CODENATION}

JS

## Time for sum...

MATHS!

{CODENATION}

JS

+

-

*

**

/

%

++

--

## Arithmetic Operators

… for **calculations**

{CODENATION}

JS

```
=
*=
+=
/=
-=
++
--
```

## Assignment Operators

... for **storing values**

{CODENATION}

JS

= Assignment Operator

## Try this...

```
let i = 10;
```

Assigning i to the number 10

## Try this...

```
let i = 10;

i = i + 2;
// i = 12
```

*Arithmetic operator

## We can do this better...

```
let i = 10;

i += 2;
// i = 12
```

*Assignment operator

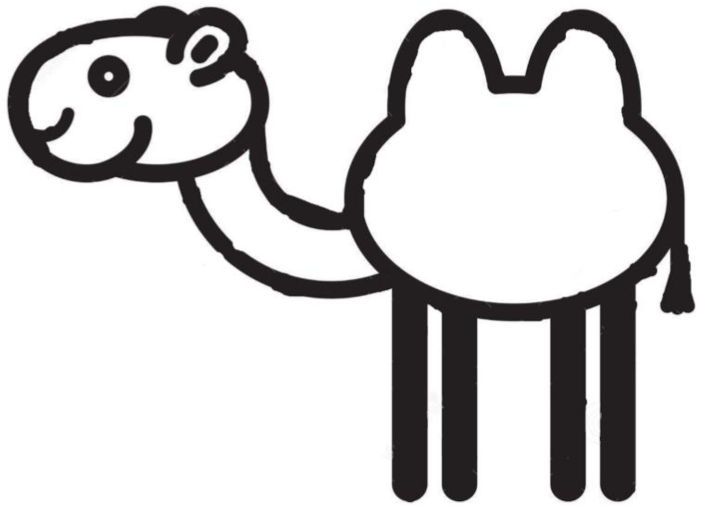
JS

## Don’t get the hump!

… introducing **camelCase**

{CODENATION}

JS



favouriteDrink
thisNumber
firstName

{CODENATION}

JS

## This is called camelCase

… it is **best practice & industry standard** as it **enhances code readability**

{CODENATION}

JS

Lets access some data in variables

## Try this...

```
let favouriteDrink = "coffee";
console.log(favouriteDrink);
```

Notice when we **console.log a variable**, we **don't need ""** like we do with a string.

## Try this...

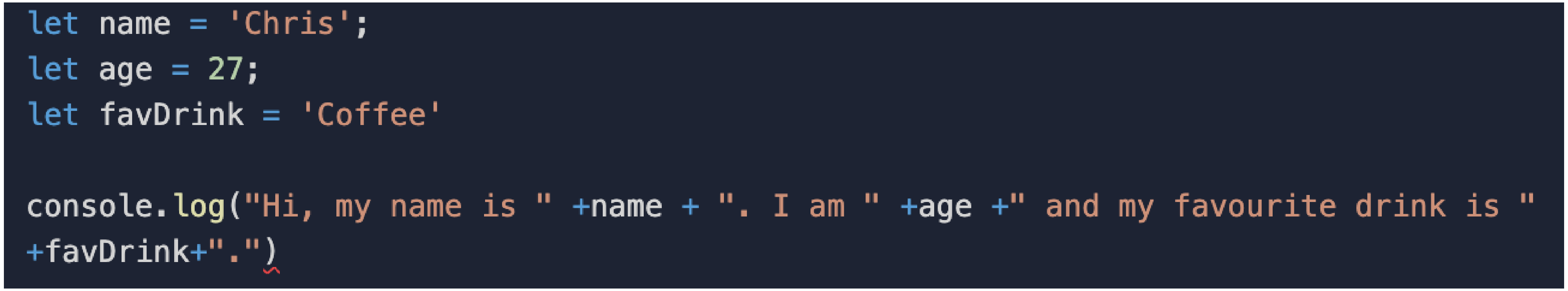
```
let favouriteDrink = "coffee";

console.log("My favourite drink
is " + favouriteDrink);
```

Putting strings together with variables is called **concatenation**. It allows us to produce sensible outputs!

{CODENATION}

## This can get messy...

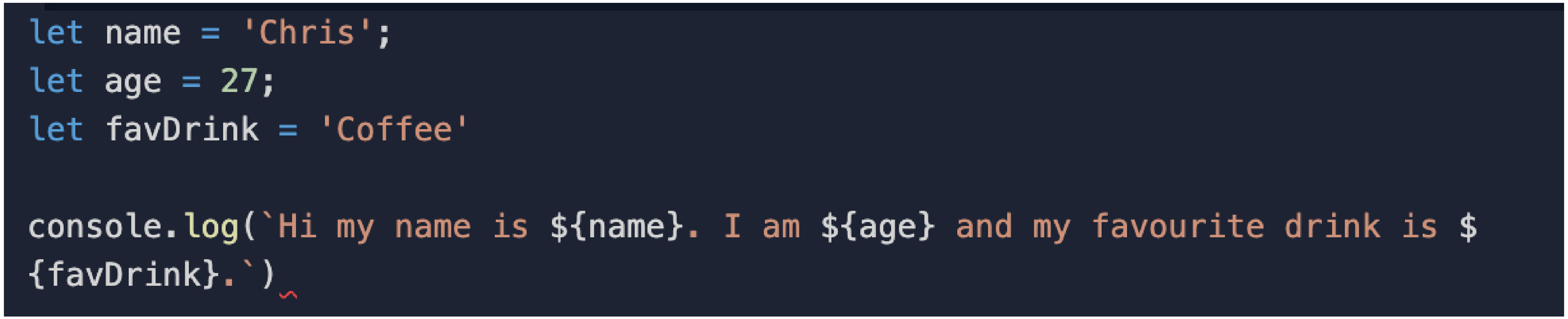


```
let name = 'Chris';
let age = 27;
let favDrink = 'Coffee'

console.log("Hi, my name is " +name + ". I am " +age +" and my favourite drink is "
+favDrink+".")
```

Using **'Template Literals'** we can inject variables into strings a lot easier

## This can get messy...

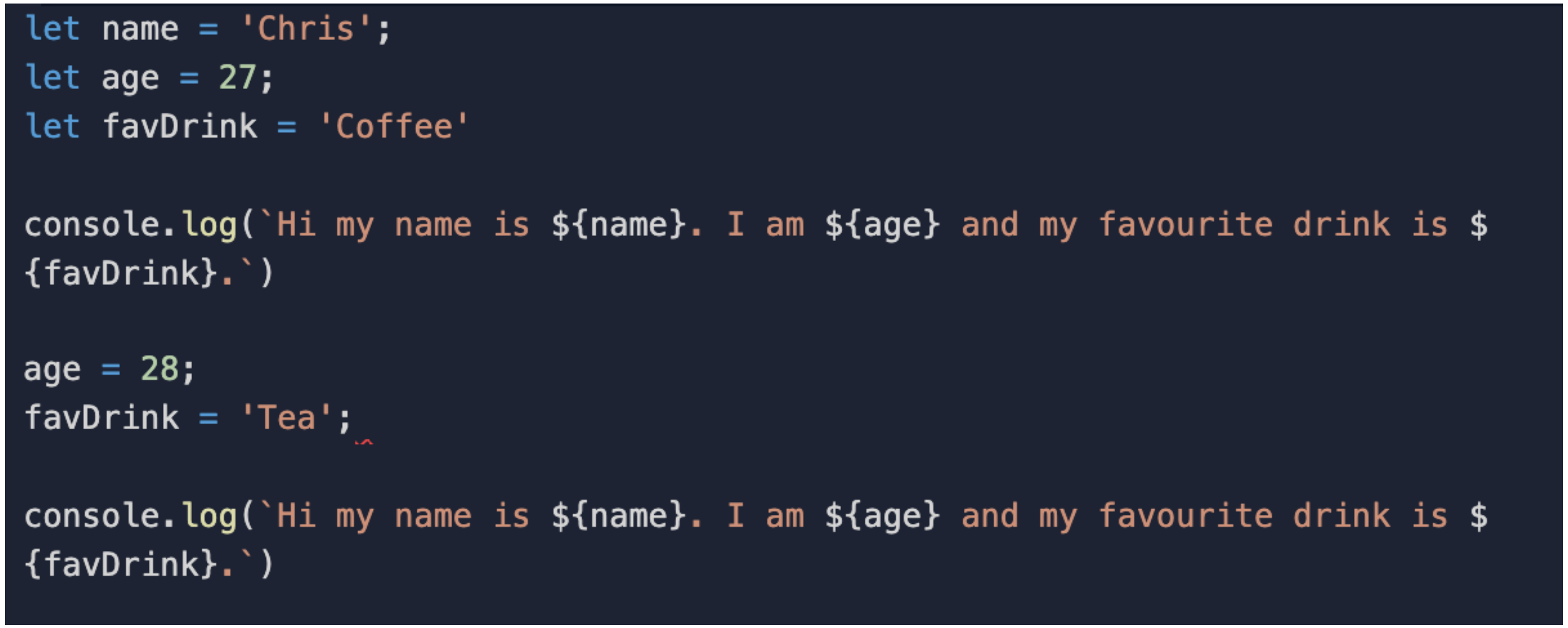


```
let name = 'Chris';
let age = 27;
let favDrink = 'Coffee'

console.log(`Hi my name is ${name}. I am ${age} and my favourite drink is $
{favDrink}.`)
```

Using **'Template Literals'** we can inject variables into strings a lot easier

## Remember


```
let name = 'Chris';
let age = 27;
let favDrink = 'Coffee'

console.log(`Hi my name is ${name}. I am ${age} and my favourite drink is $
{favDrink}.`)

age = 28;
favDrink = 'Tea';

console.log(`Hi my name is ${name}. I am ${age} and my favourite drink is $
{favDrink}.`)
```

We can also **update** our variables (if we use let)

{CODENATION}

## Learning Objectives

To understand and use variables and operators to store values and manipulate them

To use camelCase when naming variables

To understand how to access data in variables

## Activity 1:

JS

Create a program that **stores someone's name**, **age** and **favourite colour** and log it to the console in a complete sentence using **Template Literals**.

## Stretch

Update **all of your variables** and **write out a new sentence** underneath your original.

{CODENATION}

## Activity 2:

JS

Create a program that stores what you eat today for breakfast, lunch and dinner. Log these to the console.

## Stretch

Update each of these variables to what you will eat tomorrow. Log these to the console.

{CODENATION}

## Activity 3:

Create a program that **calculates the number of days** from today to your birth date.

## Hint

Look for **'Javascript Date'** on MDN

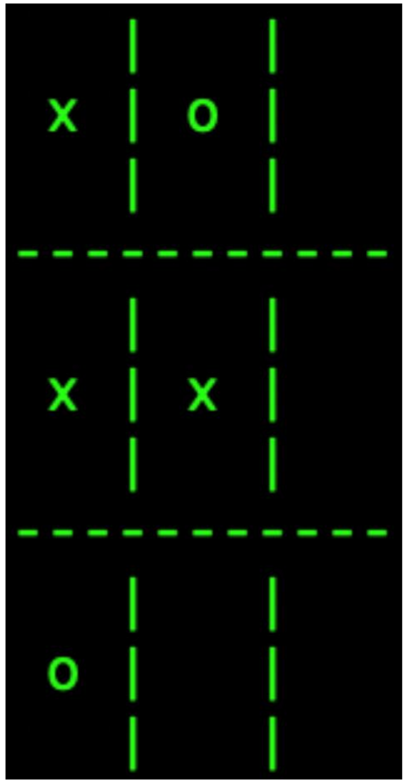
# Activity 4:

JS

> Create 9 variables: space1, space2… space9

> Assign either the value ‘x’, ‘o’,’ ’, to each of these variables.

> Insert the variables into your boards using the ${varName} syntax and make it look like the displayed board



```
     |     |
  x  |  o  |
     |     |
-----------
     |     |
  x  |  x  |
     |     |
-----------
     |     |
  o  |     |
     |     |
```

{CODENATION}

# For next time...

… take a look at **selection** and **if/else/switch.**

JS

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/if...else](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/if...else)

[https://www.youtube.com/watch?v=IsG4Xd6LlsM](https://www.youtube.com/watch?v=IsG4Xd6LlsM)

Why would we use **if/else?**
What benefit does a **"switch"** have over if/else?

{CODENATION}