# First thing's first

Create three functions that each check one of the followings:

- Types of drinks available
- Costs of drinks and remaining balance
- If the type of drink chosen is available and there's enough money left, output "You've got a good taste!"

Code Nation

# JavaScript Fundamentals

## Objects

{codenation}®

# Learning Objectives

- To understand the concept of an object
- To access data from within an object
- To use functions with objects
- To understand and use the “this” keyword


# Objects.


{cn}®
Objects are containers that
can store data and functions.


We store data inside the objects
by using key-value pairs.

```
const cafe = {
    name: "Whitesheep",
    seatingCapacity: 100,
    hasSpecialOffers: true,
    drinks: [
        "Cappuccino",
        "Latte",
        "Filter coffee",
        "Tea",
        "Hot chocolate"
    ]
};
```

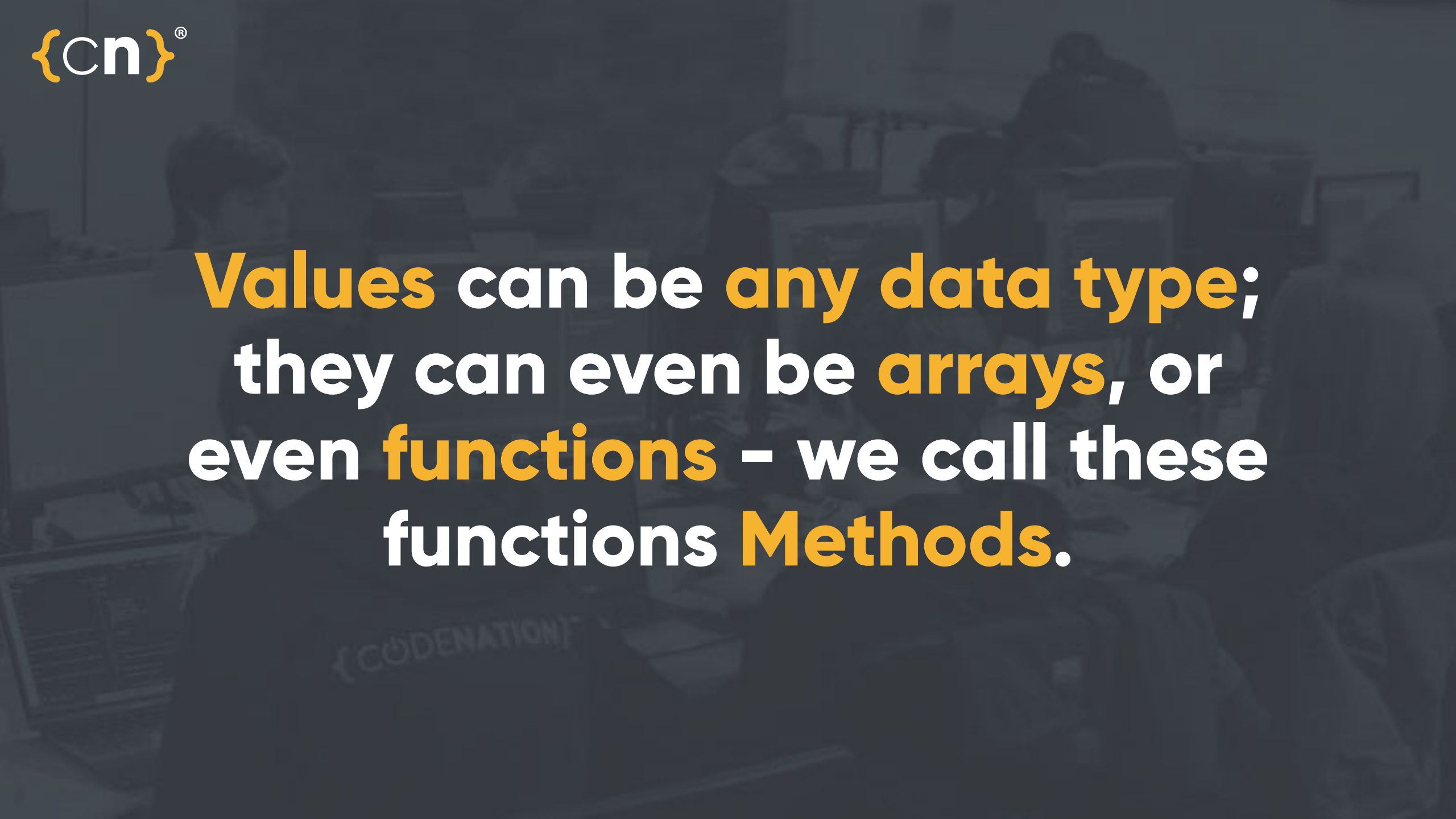
# Activity:

Let's create an object called **person** with a key called **name** and set the **value** to your name.

# Activity:

Let's create an object called **person** with a key called **name** and set the **value** to your name.

Add another **key** called **age**.


{cn}®
Values can be any data type;
they can even be arrays, or
even functions – we call these
functions Methods.


Accessing data:
object.property

```
person.name

console.log(person.name)
```


We can also use
bracket notation.

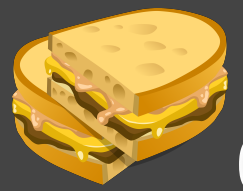
```
console.log(person["name"]);
```


Bracket notation actually
gives us a bit more flexibility.

# Back to our example:
# Let's say Whitesheep have different specials based on the time of day...

Free croissants at breakfast... 

Free drink with a sandwich at lunch...



```
let offer = "none";
let time = 1200;

const cafe = {
    name: "Whitesheep",
    seatingCapacity: 100,
    hasSpecialOffers: true,
    drinks: [
        "Cappuccino",
        "Latte",
        "Filter coffee",
        "Tea",
        "Hot chocolate"
    ],

    breakfastOffer: "Free croissant with coffee",
    lunchOffer: "Free drink with surprisingly priced sandwich",
    noOffer: "Sorry no offer"
};
```

Now that the specials are stored in the object, we can access them...

```
let offer = "none";
let time = 1200;

const cafe = {
    name: "Whitesheep",
    seatingCapacity: 100,
    hasSpecialOffers: true,
    drinks: ["Cappuccino","Latte","Filter coffee","Tea","Hot chocolate"],
    breakfastOffer: "Free croissant with coffee",
    lunchOffer: "Free drink with surprisingly priced sandwich",
    noOffer: "Sorry no offer"
};

if (time < 1100){
    offer = cafe.breakfastOffer;
    console.log(cafe.breakfastOffer);
} else if (time < 1500){
    offer = cafe.lunchOffer;
    console.log(cafe.lunchOffer);
}
```

# Activity:

Let's create an alarm.

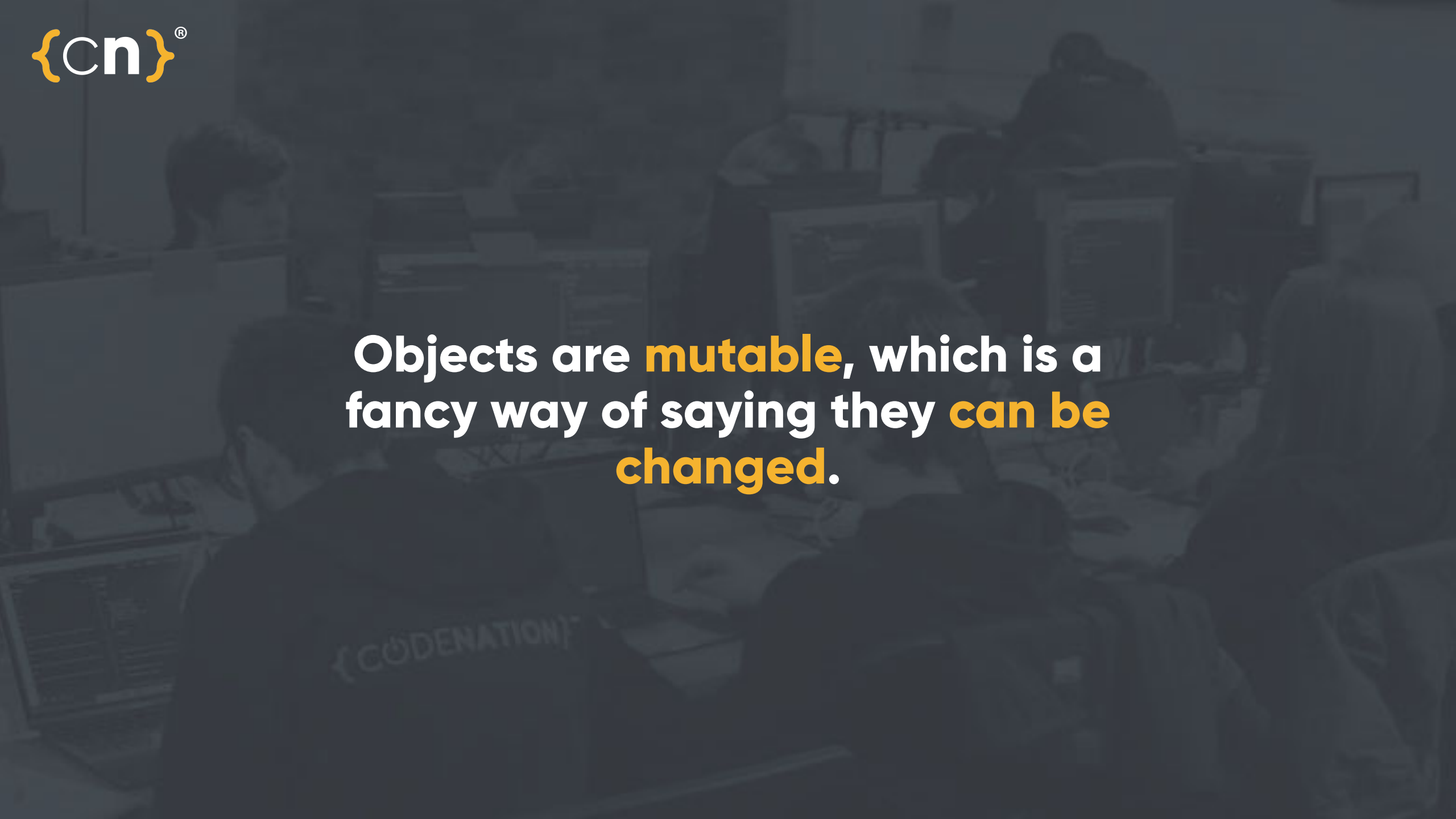
Create a key called **weekendAlarm**, with a value saying "no alarm needed", and a key called **weekdayAlarm**, with a value saying "get up at 7am"

Create a variable called day and one called alarm

If day is Saturday or Sunday, set alarm to weekendAlarm

If the day is a weekday, set alarm to weekdayAlarm


# Adding properties.


Objects are mutable, which is a fancy way of saying they can be changed.

```
cafe.biscuits = ["waffle", "shortbread"];
```

Or

```
cafe["biscuits"] = ["waffle", "shortbread"];
```

# Activity:

Let's add a list of favourite songs to our person object and log them to the console.


{cn}®
Functions* within objects.
*Remember we call these Methods

```
let offer = "none";
let time = 1200;

const cafe = {
    name: "Whitesheep",
    seatingCapacity: 100,
    hasSpecialOffers: true,
    drinks: ["Cappuccino","Latte","Filter coffee","Tea","Hot chocolate"],
    breakfastOffer: "Free croissant with coffee",
    lunchOffer: "Free drink with surprisingly priced sandwich",
    noOffer: "Sorry no offer”,

     openCafe:()=>{
         return "Come on in";
     },
     closeCafe:()=>{
         return "We are closed, come back tomorrow!"
     }
};

console.log(cafe.openCafe());
console.log(cafe.closeCafe());
```


{cn}®
ES6 made object function declaration easier...

```
openCafe:()=>{
    return "Come on in";
},
closeCafe:()=>{
    return "We are closed, come back tomorrow!"
}
```

# In ES6:

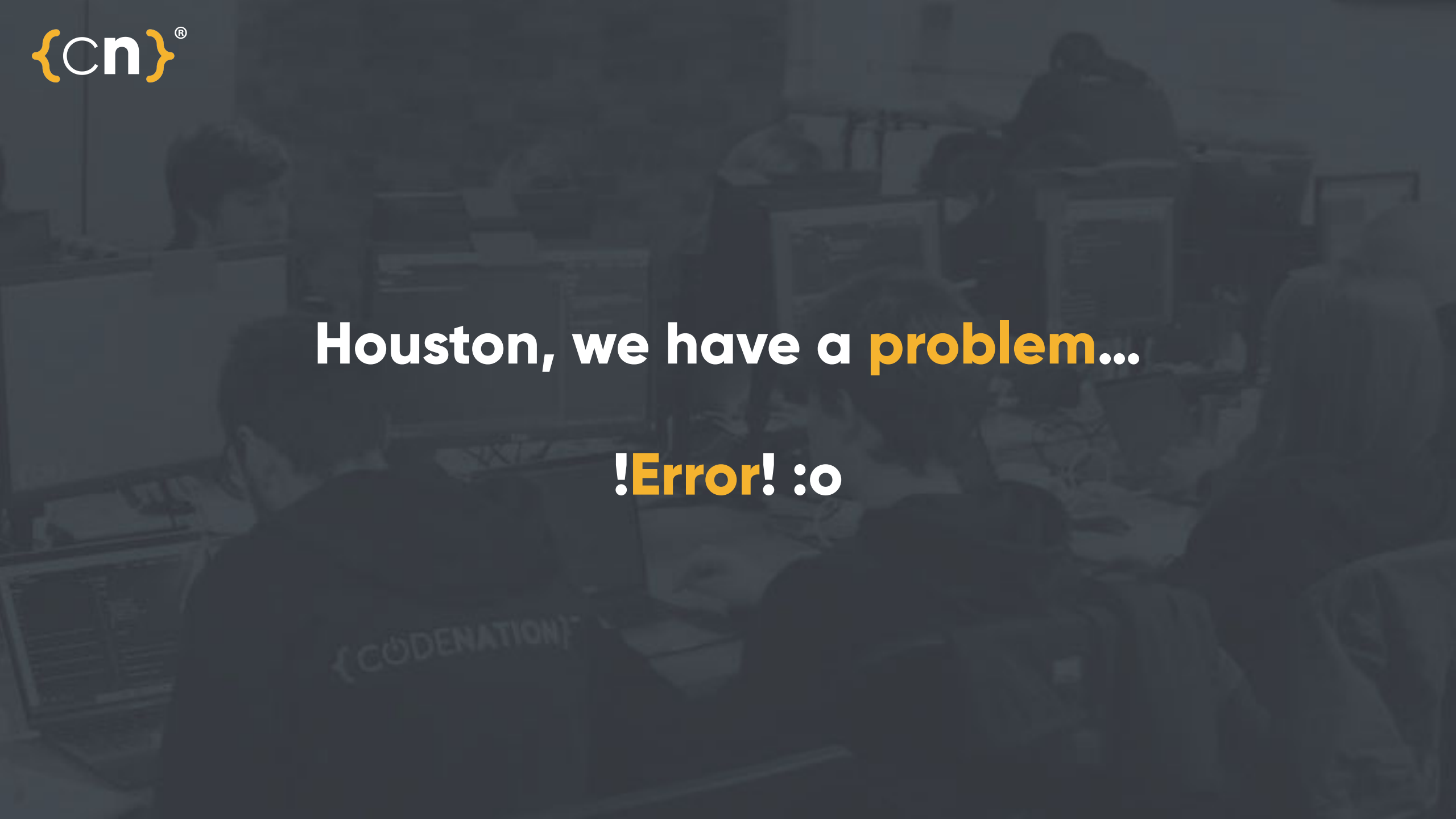
```
openCafe(){
    return "Come on in";
},
closeCafe(){
    return "We are closed, come back tomorrow!"
}
```


Methods can operate using
data from inside our objects.

```
let offer = "none";
let time = 1200;

const cafe = {
    name: "Whitesheep",
    seatingCapacity: 100,
    hasSpecialOffers: true,
    drinks: ["Cappuccino","Latte","Filter coffee","Tea","Hot chocolate"],
    breakfastOffer: "Free croissant with coffee",
    lunchOffer: "Free drink with surprisingly priced sandwich",
    noOffer: "Sorry no offer",
        openCafe(){
            if(hasSpecialOffers){
                return "Time for a special offer!";
            }
        },
    closeCafe(){
        return "We are closed, come back tomorrow!";
    }
};

console.log(cafe.openCafe());
```


Houston, we have a problem...
!Error! :o

“ReferenceError: hasSpecialOffers is not defined”…

why?

“ReferenceError: hasSpecialOffers is not defined”…

hasSpecialOffers is outside of the function’s scope…

“ReferenceError: hasSpecialOffers is not defined”…

hasSpecialOffers is outside of the function’s scope…

We need to definitively say where hasSpecialOffers is…

*Which can seem confusing on the surface*

```
let offer = "none";
let time = 1200;

const cafe = {
    name: "Whitesheep",
    seatingCapacity: 100,
    hasSpecialOffers: true,
    drinks: ["Cappuccino","Latte","Filter coffee","Tea","Hot chocolate"],
    breakfastOffer: "Free croissant with coffee",
    lunchOffer: "Free drink with surprisingly priced sandwich",
    noOffer: "Sorry no offer",
        openCafe(){
            if(hasSpecialOffers){
                return "Time for a special offer!";
            }
        },
    closeCafe(){
        return "We are closed, come back tomorrow!";
    }
};

console.log(cafe.openCafe());
```

“ReferenceError: hasSpecialOffers is not defined”…

hasSpecialOffers is outside of the function’s scope…

We need to definitively say where hasSpecialOffers is…

**To do this we use the ‘this’ keyword…**

***“hasSpecialOffers belongs to this object”***

```
let offer = "none";
let time = 1200;

const cafe = {
    name: "Whitesheep",
    seatingCapacity: 100,
    hasSpecialOffers: true,
    drinks: ["Cappuccino","Latte","Filter coffee","Tea","Hot chocolate"],
    breakfastOffer: "Free croissant with coffee",
    lunchOffer: "Free drink with surprisingly priced sandwich",
    noOffer: "Sorry no offer",
        openCafe(){
            if(this.hasSpecialOffers){
                return "Time for a special offer!";
            }
        },
    closeCafe(){
        return "We are closed, come back tomorrow!";
    }
};
console.log(cafe.openCafe());
```

# Learning Objectives

- To understand the concept of an object
- To access data from within an object
- To use functions with objects
- To understand and use the “this” keyword

# Activity:

Let's edit our person object to include...

A function called sayHi and when it's called, it should return "Hello, my name is ${this.name}"

**Refresher: Template Literal**

# Activity:

Create an object called pet with key values of:

name, typeOfPet, age, colour

And methods called eat and drink. They should return a string saying [Your pet name] is eating/drinking.

# Activity:

Create an object called coffeeShop with key values of:

branch, drinks with prices, food with prices

And methods called drinksOrdered and foodOrdered. They should return a string saying [Your order] is ... with all items chosen with costs, and the total cost.

Objectively... you're done, bud.