the Master Course

{CODENATION}

JAVASCRIPT FUNDAMENTALS

Objects

{CODENATION}

Learning Objectives

To explore the concept of an object

To access data from within an object

To use functions with objects

To discover and use the ‘this’ keyword.

Introducing Objects

... objects are containers that can store data and functions. We use **Key-Value** pairs to store the data

Let's look at one...

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

Create a **variable** called **Cafe.** The **{}** determines that this is an Object not a variable or array.

name, **seatingCapacity**, **hasSpecialOffers** and **drinks** are all **KEYS**

keys and **values** are separated by a **colon**.

key : value

{CODENATION}

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.**

Values

... can be **any data type**. They can even be **arrays** or **functions**!

Question

… how do you think we **access data** in an **object**?

JS

object.property

```
person.name
console.log(person.name)
```

{CODENATION}

But that’s not all...

... we can also use **bracket notation**

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

JS

`person.name` VS `person["name"]`

Both common and worth knowing!

{CODENATION}

With brackets...

... we can use **variables** to select the **keys** of an object!

Let's say

… whitesheep may have different specials **based on the time of day**.

Free croissants at breakfast...

Free drink with a sandwich at lunch...

{CODENATION}

```javascript
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"
};
```

We could

… put each **special in an object** and select one at a **specific time**.

```
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 day is a weekday, set alarm to **weekdayAlarm.**

Objects are mutable

… which is a posh way of saying **we can change them** once we've made them.

```
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.

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

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

{CODENATION}

Since ES6, a modern version of Javascript its easier to declare functions in objects.

You don't need the colon, nor the arrow syntax to create functions.

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

=> functions are currently industry standard and impact scope (week 4).

In ES6:

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

ES6 is the newest version of Javascript.

{CODENATION}

So, let's push functions a **little further** and have them operate on data within our 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(hasSpecialOffers){
            return "Time for a special offer!";
        }
    },
    closeCafe(){
        return "We are closed, come back tomorrow!";
    }
};

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

{CODENATION}

!Error!

hasSpecialOffers is actually **outside** of the functions **scope**.

We need to tell openCafe **where** hasSpecialOffers is.

We do this using the **this** keyword.

```
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());
```

{CODENATION}

this

... means this current object.

So accessing this.hasSpecialOffers

... inside the object is the same as saying cafe.hasSpecialOffers outside of it.

Learning Objectives

To explore the concept of an object

To access data from within an object

To use functions with objects

To discover and use the ‘this’ keyword.

Activity 1:

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}"**

Activity 2:

Create an **object** called pet with the 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 3:

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 total costs.

Further Reading...

... take a look at **HTML.**

https://developer.mozilla.org/en-US/docs/Web/HTML

https://www.youtube.com/watch?v=u0OeZfIfBRI

Can you name any **HTML Elements?**

Before the next lecture I want you to research **at least THREE!**