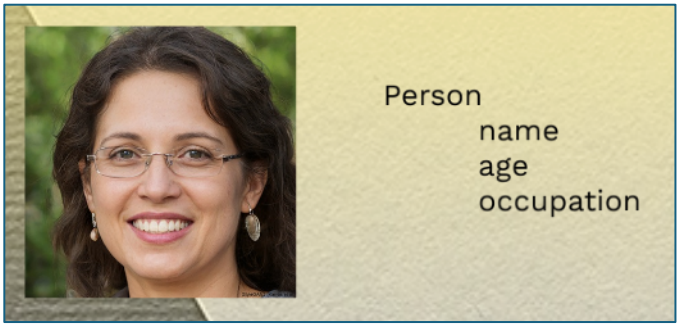
Objects

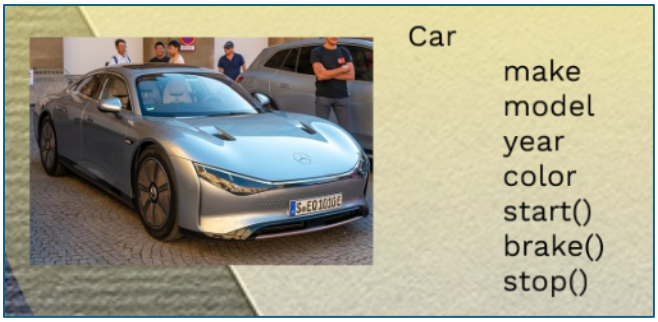
Objects are collections of related data. They allow you to group values together to represent real-world entities and their attributes. Objects are essential when working with APIs, which play a big role in dynamic web development. In fact, objects use a structure similar to the data you'll commonly receive from APIs.

For example, we might want to store information about a course that includes different details like the course title, number of credits, and description.

Or, we might want to represent people, along with their name, age, and occupation.



Another example could be a car. In addition to having descriptive properties like make, model, year, and color, a car object could also include **methods**—actions that the object can perform—such as starting, braking, or stopping.



Properties

Objects store data using **key/value pairs**. In the example below, the 'person' object has three key/value pairs:

```
let person = {
     name: 'Jacob',
     age: 25,
     job: 'Developer'
};
```

Here:

- 'name' is the key, and 'Jacob' is the value.
- 'age' is the key, and 25 is the value.
- 'job' is the key, and 'Developer' is the value.

To access these values in code, we use **dot notation**, referencing the object name followed by the property name:

person.name returns 'Jacob'
person.age returns 25.

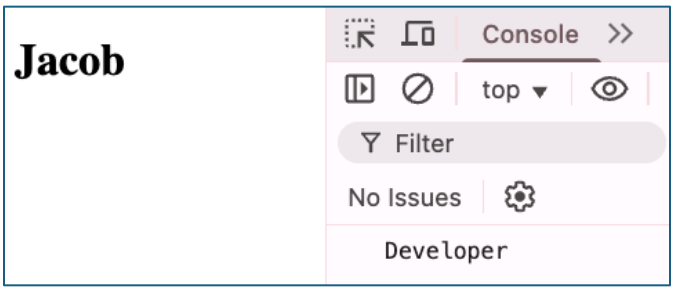
These values can be used just like any other variable:

```
let person = {
    name: 'Jacob',
    age: 25,
    job: 'Developer'
};

console.log(person.job);

let title = document.querySelector('h1');

title.textContent = person.name;
```



You've already been working with objects and their properties! For instance:

```
element.textContent

inputElement.value
```

These are properties of **DOM objects** like elements and form inputs.

Methods

Objects can also include **methods**, which are functions stored as object properties. Here's an example of a car object with a method:

```
let car = {
    brand: "Honda",
    model: "Passport",
    start: function() {
      console.log("Car is starting...");
    }
  };

car.start();
```

You've seen methods before:

```
document.querySelector('h1')

document.createElement('img')

console.log('text for console')
```

These are built-in objects (like 'document' and 'console') that come with their own properties and methods. While we can't modify those, we can now build our own objects with custom properties and methods.

The 'this' Keyword

Take a look at the following 'product' object:

```
let product = {
  id: 101,
```

```
  name: "Lamp",
  price: 9.99,
  quantity: 2,
  getTotal: function() {
    return this.price * this.quantity;
  }
};
```

In this case, the getTotal method uses the ‘this’ keyword. The ‘this’ keyword refers to the object itself. So, inside the product object:

- this.price refers to product.price
- this.quantity refers to product.quantity

```
let total = product.getTotal();
```

Would result in total being assigned 19.98.

So, there we have JavaScript objects used to represent real-world entities, storing data in properties and actions in methods, which we can reference and use in our code.