JS Objects

Lesson Time: 30 Minutes

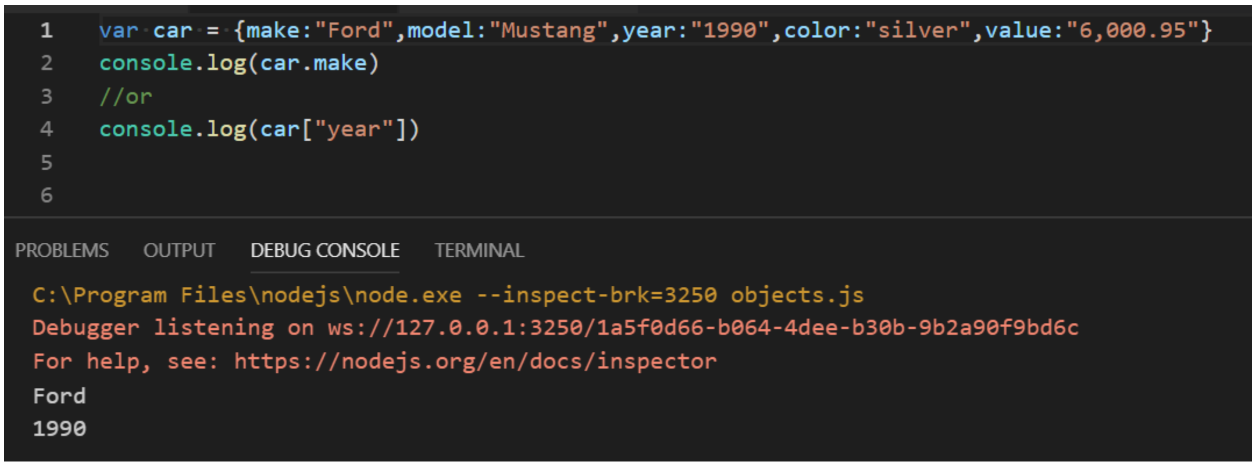
Objects Are Complex Data

So far we've looked at basic forms of data in Javascript--string text, numbers, booleans, and the undefined data type. These data types are called primitive data types. When we want to create more complex data, we can store in a JS object.

A JS object is made up of key/value pairs. In other languages, such as Python, this is also known as a data dictionary or hashtable, but in JS, we call them objects.
Here's what they looks like:

```
{keyname : “value”, keyname2: “value”, keyname3: “value”...}
```

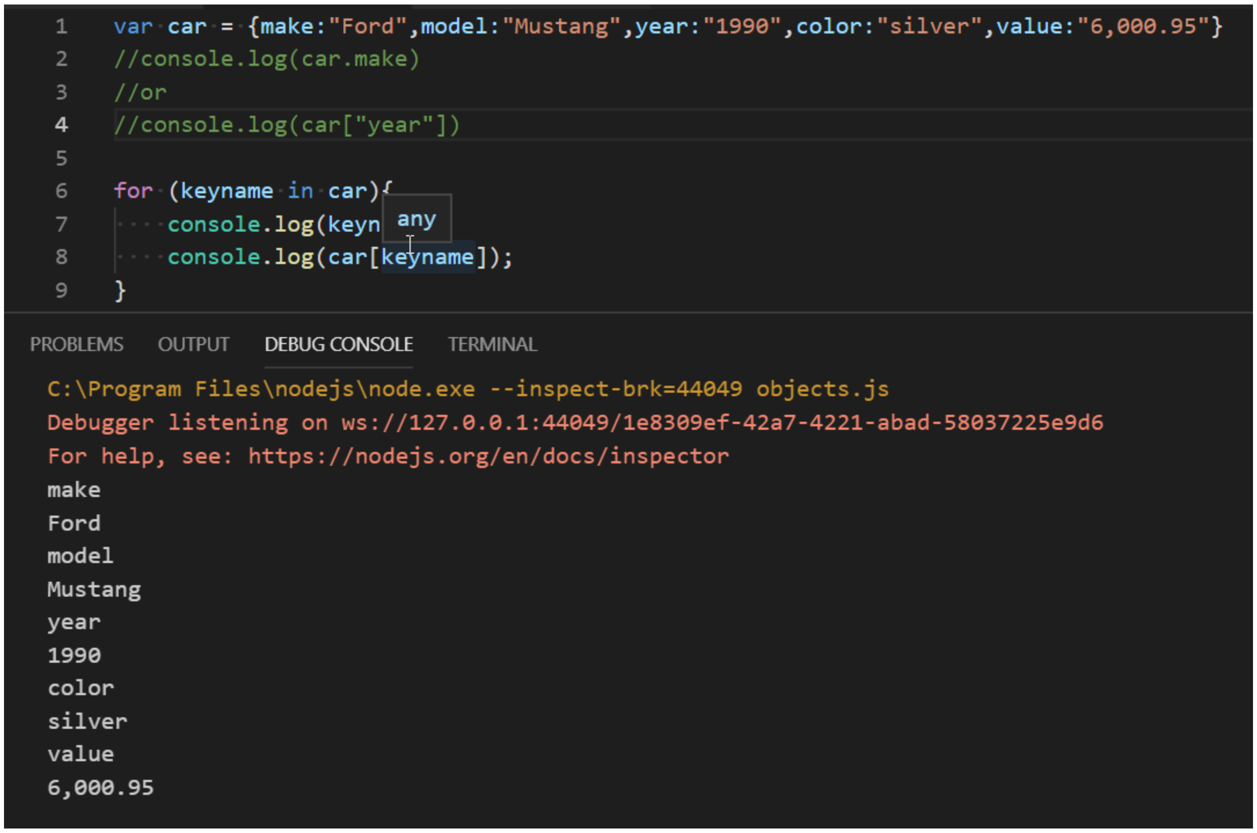
Below is an example in code. If we want to represent data about a car for example, a JS object is a great way to do it, because a car is more complex than just a number, date, or text.



```
var car = {make:"Ford",model:"Mustang",year:"1990",color:"silver",value:"6,000.95"}
console.log(car.make)
//or
console.log(car["year"])
```

```
C:\Program Files\nodejs\node.exe --inspect-brk=3250 objects.js
Debugger listening on ws://127.0.0.1:3250/1a5f0d66-b064-4dee-b30b-9b2a90f9bd6c
For help, see: https://nodejs.org/en/docs/inspector
Ford
1990
```

We can loop through the properties of an object with a for...in loop

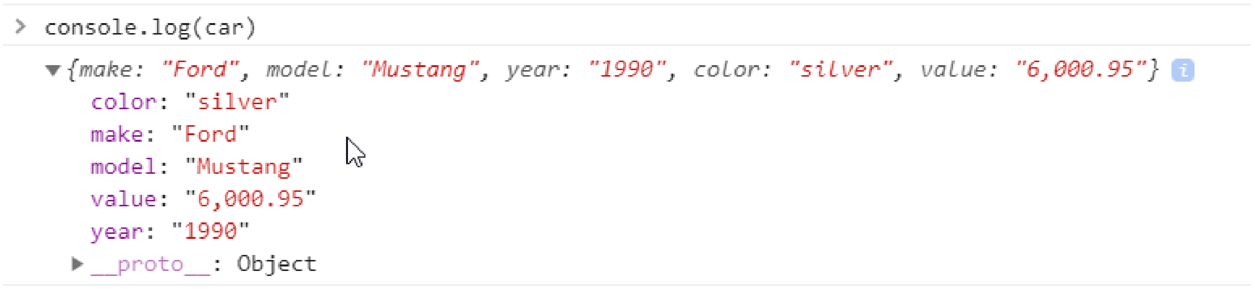


```
var car = {make:"Ford",model:"Mustang",year:"1990",color:"silver",value:"6,000.95"}
//console.log(car.make)
//or
//console.log(car["year"])

for (keyname in car){
    console.log(keyname);
    console.log(car[keyname]);
}
```

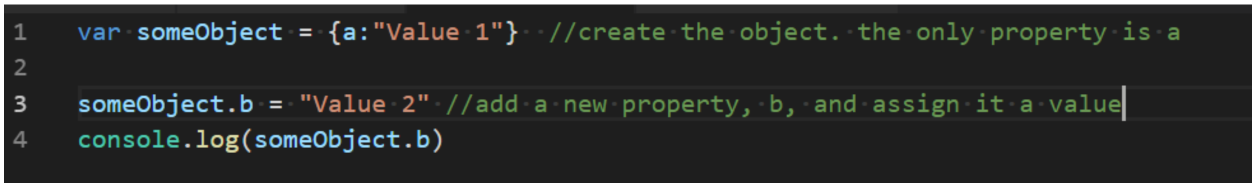
```
C:\Program Files\nodejs\node.exe --inspect-brk=44049 objects.js
Debugger listening on ws://127.0.0.1:44049/1e8309ef-42a7-4221-abad-58037225e9d6
For help, see: https://nodejs.org/en/docs/inspector
make
Ford
model
Mustang
year
1990
color
silver
value
6,000.95
```

A screenshot of the object from Chrome Developer Console



```
> console.log(car)
  ▼{make: "Ford", model: "Mustang", year: "1990", color: "silver", value: "6,000.95"}
      color: "silver"
      make: "Ford"
      model: "Mustang"
      value: "6,000.95"
      year: "1990"
    ▶ __proto__: Object
```

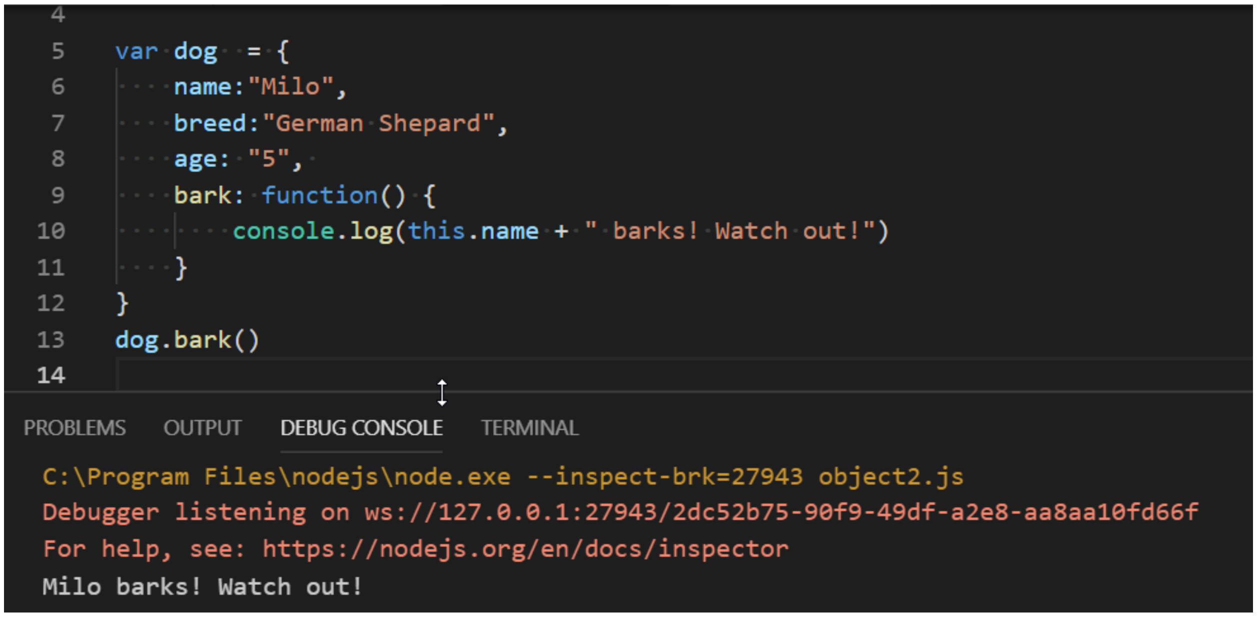
We can add onto an object after it’s been created.



```
var someObject = {a:"Value 1"}  //create the object. the only property is a

someObject.b = "Value 2" //add a new property, b, and assign it a value
console.log(someObject.b)
```

One last important feature of objects is that they can contain a function. When a function is stored inside of an object, we call it an **object method**. Here’s our example.



```
var dog  = {
    name:"Milo",
    breed:"German Shepard",
    age: "5",
    bark: function() {
        console.log(this.name + " barks! Watch out!")
    }
}
dog.bark()
```

```
C:\Program Files\nodejs\node.exe --inspect-brk=27943 object2.js
Debugger listening on ws://127.0.0.1:27943/2dc52b75-90f9-49df-a2e8-aa8aa10fd66f
For help, see: https://nodejs.org/en/docs/inspector
Milo barks! Watch out!
```

Lastly, in the code example above, you will notice the keyword **this**. This references the name property of the object itself, in this case, milo the dog.

In conclusion, we use javascript objects to group together related data into one unit. Doing this allows us to create complex units of data like cars, dogs, and everything else.

To learn more about objects, check out the reference guide on the Mozilla Developer site.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object

Key Terms	JS Object

Lesson Files	
Additional Resources	https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object
Further Learning	

LAB: JS Objects

Lesson Time: 60 Minutes

In this lab, you'll start creating some complex data and storing in it objects.

1. Create an object called movieStar. The movieStar object will have the following data:
   - firstname
   - lastname
   - birthday
   - weight
   - gender
   - eyecolor
   - haircolor

2. Create an object called movie. The movie object will have the following data;
   - moviename
   - Yearreleased
   - moviestar

HINT (the movieStar object you created in step one is a part of the data of the 2nd object!)
HINT2 (JS is case sensitive!)

3. Log the movieStar's birthday to the console.
4. Log the Movie's moviestar to the console.
5. Log ONLY the firstname of the movie's movestar to the console.

6. Create an object called boat. The boat object will have the following data;

- boatname
- color
- size
- A method called SetSail.
    - The method accept 1 movieStar object as argument.
    - The method will not return any data
    - The method will print to the console “moviestar’s firstname + lastname sets sail on the boat!”

7. Call the setSail method on the boat and pass the movieStar to it.