JavaScript Essentials

Objects

Table of Contents

Lesson Objectives

- Understand the fundamental JavaScript object syntax
- Understand the basic theory behind object-oriented programming, how this relates to JavaScript ("most things are objects"), and how to start working with JavaScript objects
- Able to access and setting member in Object using Dot notation and Bracket notation

Section 1

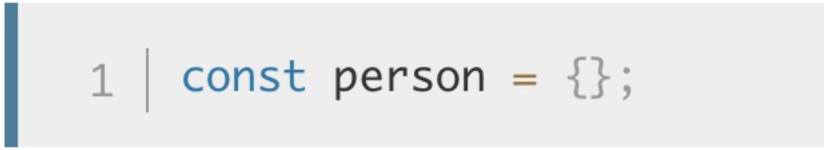
Overview – Object basics

Overview – Object basics

- An object is a collection of related data and/or functionality (which usually consists of several variables and functions — which are called properties and methods when they are inside objects.)
- Let's work through an example to understand what they look like.

Overview – Object basics

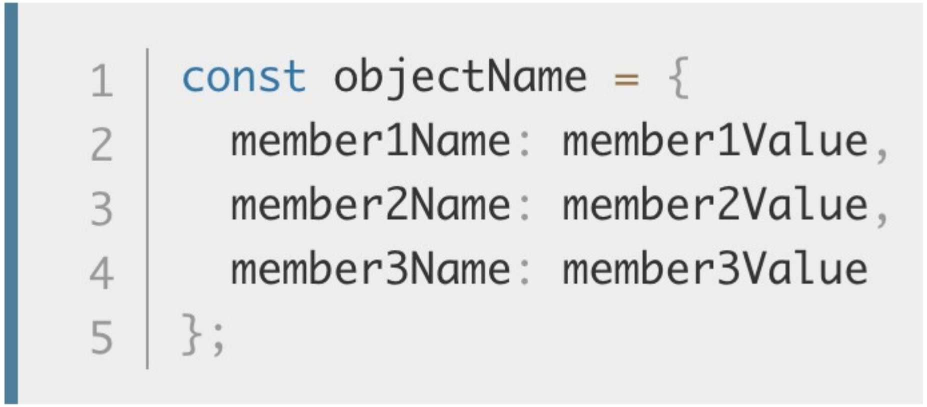
- To begin with, make a local copy of our oojs.html file. This contains very little — a <script> element for us to write our source code into. We'll use this as a basis for exploring basic object syntax. While working with this example you should have your developer tools JavaScript console open and ready to type in some commands.
- As with many things in JavaScript, creating an object often begins with defining and initializing a variable. Try entering the following line below the JavaScript code that's already in your file, then saving and refreshing:



```
const person = {};
```

Overview – Object basics

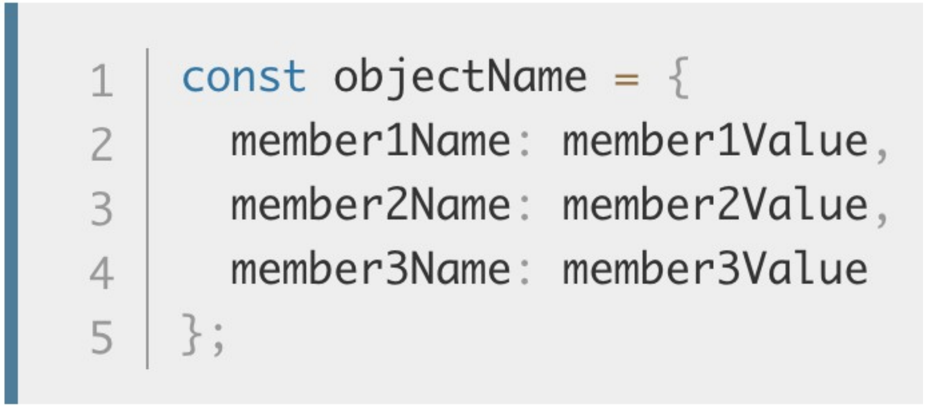
- So what is going on here? Well, an object is made up of multiple members, each of which has a name (e.g. name and age above), and a value (e.g. ['Bob', 'Smith'] and 32). Each name/value pair must be separated by a comma, and the name and value in each case are separated by a colon. The syntax always follows this pattern:



```
const objectName = {
  member1Name: member1Value,
  member2Name: member2Value,
  member3Name: member3Value
};
```

Overview – Object basics

- So what is going on here? Well, an object is made up of multiple members, each of which has a name (e.g. name and age above), and a value (e.g. ['Bob', 'Smith'] and 32). Each name/value pair must be separated by a comma, and the name and value in each case are separated by a colon. The syntax always follows this pattern:



```
const objectName = {
  member1Name: member1Value,
  member2Name: member2Value,
  member3Name: member3Value
};
```

Overview – Object basics

- The value of an object member can be pretty much anything — in our person object we've got a string, a number, two arrays, and two functions.
- The first four items are data items, and are referred to as the object's **properties**.
- The last two items are functions that allow the object to do something with that data, and are referred to as the object's **methods**

Overview – Object basics

- An object like this is referred to as an **object literal** — we've literally written out the object contents as we've come to create it.
- This is in contrast to objects instantiated from classes, which we'll look at later on.
- It is very common to create an object using an object literal when you want to transfer a series of structured, related data items in some manner, for example sending a request to the server to be put into a database.
- Sending a single object is much more efficient than sending several items individually, and it is easier to work with than an array, when you want to identify individual items by name.

Overview – Summary

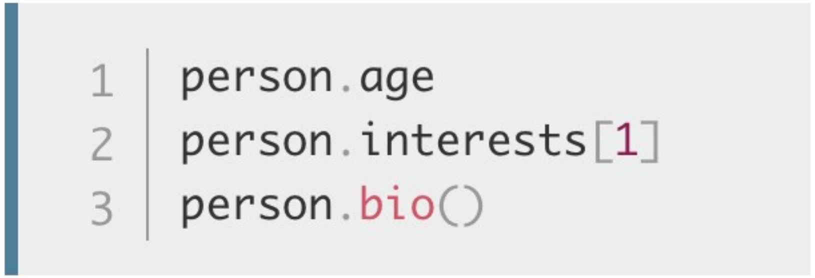
- An object is a collection of related data and/or functionality (which usually consists of several variables and functions — which are called properties and methods when they are inside objects.)
- The value of an object member can be pretty much anything
- If it’s a value we refer to it as a **property**
- If it’s a function we refer to it as a **method**
- Object literal - we've literally written out the object contents as we've come to create it
- Object is much more efficient than sending several items individually, and it is easier to work with than an array, when you want to identify individual items by name

Section 2

Dot notation

Dot notation – Basic syntax

- Above, you accessed the object's properties and methods using **dot notation**. The object name (person) acts as the **namespace** — it must be entered first to access anything **encapsulated** inside the object
- Next you write a dot, then the item you want to access — this can be the name of a simple property, an item of an array property, or a call to one of the object's methods, for example:



```
person.age
person.interests[1]
person.bio()
```

Dot notation – Sub namespace

- It is even possible to make the value of an object member another object. For example, try changing the name member

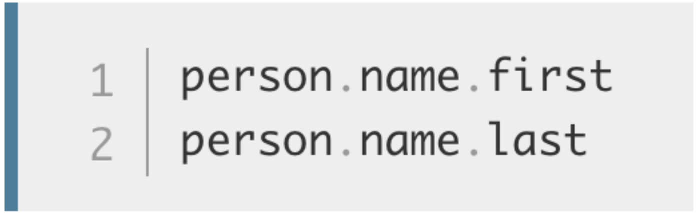
from

name: ['Bob', 'Smith’],
to

name : { first: 'Bob', last: 'Smith' },

Dot notation – Sub namespace

- Here we are effectively creating a **sub-namespace**. This sounds complex, but really it's not — to access these items you just need to chain the extra step onto the end with another dot. Try these in the JS console:



```
person.name.first
person.name.last
```

Dot notation - Summary

- To access member of an object use Dot nation
- The object name acts as the **namespace** — it must be entered first to access anything **encapsulated** inside the object
- We can create **sub-namespace** by setting value of a member to another object

Section 3

Bracket notation

Bracket notation

- There is another way to access object properties — using bracket notation. Instead of using these:

```
person.age
person.name.first
```

- You can use

```
person['age']
person['name']['first']
```

Bracket notation

- With this notation, you can use variable to access a member
- This looks very similar to how you access the items in an array, and it is basically the same thing — instead of using an index number to select an item, you are using the name associated with each member's value.
- It is no wonder that objects are sometimes called **associative arrays** — they map strings to values in the same way that arrays map numbers to values.

Bracket notation – Summary

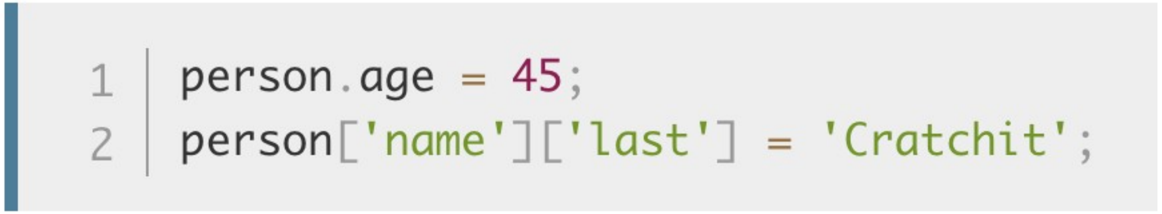
- Use Bracket notation to access a member using either ‘string’ or a variable
- If it’s a variable inside Bracket, then JavaScript will resolve the value of that variable and access the member of corresponding key
- Objects are sometimes called **associative arrays**

Section 4

Setting object members

Setting object members

- So far we've only looked at retrieving (or getting) object members — you can also set (update) the value of object members by simply declaring the member you want to set (using dot or bracket notation), like this:



```
person.age = 45;
person['name']['last'] = 'Cratchit';
```

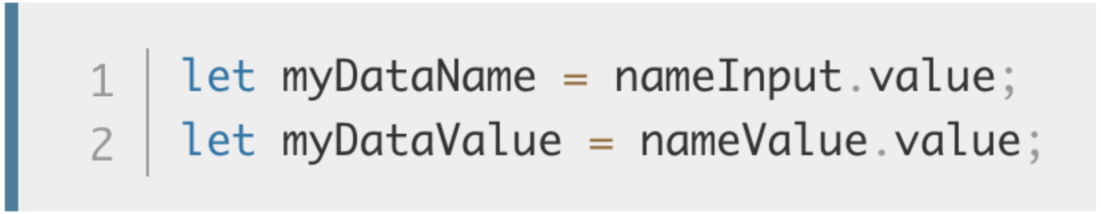
Setting object members

- Setting members doesn't just stop at updating the values of existing properties and methods; you can also create completely new members. Try these in the JS console:

```
person['eyes'] = 'hazel';
person.farewell = function() { alert("Bye everybody!"); }
```

Setting object members

- One useful aspect of bracket notation is that it can be used to set not only member values dynamically, but member names too.
- Let's say we wanted users to be able to store custom value types in their people data, by typing the member name and value into two text inputs. We could get those values like this:



```
let myDataName = nameInput.value;
let myDataValue = nameValue.value;
```

Setting object members

- We could then add this new member name and value to the person object like this:

```
person[myDataName] = myDataValue;
```

- To test this, try adding the following lines into your code, just below the closing curly brace of the person object:

```
let myDataName = 'height';
let myDataValue = '1.75m';
person[myDataName] = myDataValue;
```

Setting object members

- Now try saving and refreshing, and entering the following into your text input:

```
person.height
```

- Adding a property to an object using the method above isn't possible with dot notation, which can only accept a literal member name, not a variable value pointing to a name.

Setting object members - Practice Time

Practice Setting object members

Setting object members – Summary

- To setting object members you can use either dot notation or bracket notation
- Then use assignment (=) to set value
- If the memberKey does exists then the value is updated
- If the memberKey doesn’t exists, then a new member will be created

Objects – Summary

- As you've been going through these examples, you have probably been thinking that the dot notation you've been using is very familiar.
- That's because you've been using it throughout the course!
- Every time we've been working through an example that uses a built-in browser API or JavaScript object, we've been using objects, because such features are built using exactly the same kind of object structures that we've been looking at here, albeit more complex ones than in our own basic custom examples.

Objects – Summary

- So when you used string methods like:

```
myString.split(',');
```

- You were using a method available on an instance of the String class. Every time you create a string in your code, that string is automatically created as an instance of String, and therefore has several common methods and properties available on it.

Objects – Summary

- Should now have a good idea of how to work with objects in JavaScript — including creating your own simple objects.
- Objects are very useful as structures for storing related data and functionality
- If you tried to keep track of all the properties and methods in our person object as separate variables and functions, it would be inefficient and frustrating, and we'd run the risk of clashing with other variables and functions that have the same names.
- Objects let us keep the information safely locked away in their own package, out of harm's way.

Thank you

Q&A