4a Theory JS API pages 63 - 68 Objects

Nicholas Bruzzese

July 25, 2025

1 Introduction

Today, we are exploring the concept of **objects** in JavaScript. Objects are a powerful way to store and manage information, similar to arrays but with some key differences. This lecture is designed for young coders who have some experience with JavaScript and want to understand how objects work. We will cover what objects are, how to create and use them, and why they are useful, keeping explanations clear and simple.

2 What Are JavaScript Objects?

In JavaScript, objects are used to describe a single thing with multiple characteristics. Unlike arrays, which are great for lists (like a list of animal names), objects are perfect for storing details about one thing, like a pet cat. An object is a collection of **key-value pairs**, where:

- **Keys** are labels (strings) that identify pieces of information.
- **Values** are the data tied to those labels.

For example, to describe a cat named Harmony, we might use keys like `name`, `legs`, and `colour`, with values like `"Harmony"`, `3`, and `"Tortoiseshell"`. These key-value pairs group all the information together neatly.

3 Creating Objects

To create an object in JavaScript, we use **curly brackets `{}`**. Inside, we list key-value pairs, with a colon (`:`) between each key and value, and commas (`,`) to separate pairs. Here is an example:

```
    var cat = {
      legs: 3,
      name: "Harmony",
      colour: "Tortoiseshell"
    };

```

Breaking this down:

- `var cat` creates a variable to hold the object.

- Curly brackets `{}` define the object.
- Each key-value pair has a key (e.g., `legs`), a colon, and a value (e.g., `3`).
- Commas separate pairs, but the last pair does not need a comma.

You can also write the object on one line:

```
var cat = { legs: 3, name: "Harmony", colour: "Tortoiseshell"
};
```

However, separate lines are easier to read for longer objects.

4 Accessing Values in Objects

To retrieve information from an object, we use two methods: **square bracket notation** and **dot notation**.

4.1 Square Bracket Notation

This method uses square brackets with the key as a string, similar to accessing array elements but using a key instead of an index:

```
cat["name"]; // Returns "Harmony"
```

The key is enclosed in quotes inside square brackets. This works for any key, even those with spaces or special characters.

4.2 Dot Notation

Dot notation is a simpler way to access values, using a dot (`.`) followed by the key without quotes:

```
cat.name; // Returns "Harmony"
```

Dot notation is cleaner but only works for keys that are simple words without spaces or special characters. For example, `cat["favourite food"]` would be needed for a key with a space.

5 Getting a List of All Keys

To see all the keys in an object, we use the `Object.keys()` method, which returns an array of the object's keys. For example:

```
var dog = {
  name: "Pancake",
  age: 6,
  colour: "White",
  bark: "Yip yap!"
};
```

```

var cat = {
  name: "Harmony",
  age: 8,
  colour: "Tortoiseshell"
};

Object.keys(dog); // Returns ["name", "age", "colour", "bark"]
Object.keys(cat); // Returns ["name", "age", "colour"]

```

This is useful for checking what information an object contains or looping through its keys.

6 Adding Values to Objects

You can add key-value pairs to an object after creating it, similar to adding items to an array. You can start with an empty object:

```
var cat = {};

```

Then add key-value pairs using either square bracket notation or dot notation.

6.1 Using Square Bracket Notation

Assign a value to a key using square brackets:

```
cat["legs"] = 3;
cat["name"] = "Harmony";
cat["colour"] = "Tortoiseshell";

```

This creates an object with those key-value pairs.

6.2 Using Dot Notation

Alternatively, use dot notation for simple keys:

```
cat.legs = 3;
cat.name = "Harmony";
cat.colour = "Tortoiseshell";

```

Both methods produce the same result, but dot notation is quicker for keys without spaces.

7 What Happens If a Key Doesn't Exist?

If you try to access a key that isn't in the object, JavaScript returns `undefined`, meaning "nothing is here." For example:

```
var dog = {
  name: "Pancake",
  legs: 4,
  isAwesome: true
};

dog.isBrown; // Returns undefined
```

This helps you check if a key exists in an object.

8 Objects Don't Have a Specific Order

Unlike arrays, where elements have a clear order (index 0 before index 1), object keys have **no specific order**. JavaScript does not guarantee the order of keys, so the same object might appear differently in different browsers:

```
{ legs: 3, name: "Harmony", colour: "Tortoiseshell" }
// or
{ name: "Harmony", colour: "Tortoiseshell", legs: 3 }
```

Never write code that depends on keys being in a specific order. Use arrays if order matters.

9 Why Are Objects Useful?

Objects group related information together, making your code cleaner. Instead of separate variables:

```
var catName = "Harmony";
var catLegs = 3;
var catColour = "Tortoiseshell";
```

You can use one object:

```
var cat = {
  name: "Harmony",
  legs: 3,
  colour: "Tortoiseshell"
};
```

This is especially helpful for describing complex things, like characters in a game or users on a website.

10 Recap

To summarise:

1. Objects store information about one thing using **key-value pairs**.
2. Create objects with curly brackets `{}` and key-value pairs separated by commas.
3. Access values using `object["key"]` (square brackets) or `object.key` (dot notation).
4. Use `Object.keys(object)` to get an array of all keys.
5. Add key-value pairs with square brackets or dot notation.
6. Accessing a non-existent key returns `undefined`.
7. Object keys have no fixed order, so don't rely on their sequence.

11 Practice Time

Try these in your JavaScript console:

1. Create an object for your favourite animal with at least three key-value pairs.
2. Access one value using both square bracket and dot notation.
3. Add a new key-value pair.
4. Use `Object.keys()` to list all keys.

Example:

```
  var myPet = {
    name: "Fluffy",
    age: 2,
    type: "Rabbit"
  };

  console.log(myPet["name"]); // Prints "Fluffy"
  console.log(myPet.age); // Prints 2
  myPet.colour = "White";
  console.log(Object.keys(myPet)); // Prints ["name", "age", "type", "colour"]
```

12 Conclusion

Objects are a versatile tool in JavaScript for organising information. By using key-value pairs, you can describe anything in detail, making your code more efficient and readable. Practice using objects in your projects, and you'll see how powerful they are. If you have questions or want more examples, feel free to ask!