4b Theory JS API pages 63 - 75 Combining Arrays & Objects

Nicholas Bruzzese

July 26, 2025

Today, we're studying hwo to combining arrays and objects in JavaScript. You've worked with arrays and objects before, but did you know you can mix them together to create powerful ways to store and organise data? This lecture will explain how arrays and objects can work together, using simple examples to make it clear. We'll cover arrays of objects, accessing nested values, exploring objects in the Chrome console, using objects to link strings and values, and storing information in a structured way, like a movie collection. Let's get started!

1 Arrays of Objects

In JavaScript, arrays and objects are like containers for storing information. Arrays hold a list of items, and objects hold key-value pairs. But here's the cool part: you can put objects *inside* an array! This is called an **array of objects**, and it's a great way to group related information.

Imagine you're building a dinosaur database. You could create an array where each item is an object containing details about a dinosaur. For example:

```
var dinosaurs = [
{ name: "Tyrannosaurus Rex", period: "Late Cretaceous" },
{ name: "Triceratops", period: "Late Cretaceous" },
{ name: "Velociraptor", period: "Late Cretaceous" }
];

```

Here, `dinosaurs` is an array, and each item in the array is an object with two keys: `name` and `period`. The `name` key holds the dinosaur's name (a string), and the `period` key holds the time period it lived in (also a string). This structure lets you store lots of dinosaurs in one place, and each dinosaur has its own set of details. You can think of it like a list of dinosaur fact sheets, where each fact sheet (object) is stored in a filing cabinet (array).

2 Accessing Nested Values

Now that you have an array of objects, how do you get information out of it? This is where **accessing nested values** comes in. Since arrays use numbers (indices) to access

items and objects use strings (keys), you can combine these to dig into the data.

Let's use another example: an array of friends, where each friend is an object with details like their name, age, and lucky numbers. Here's how it might look:

```
var anna = { name: "Anna", age: 11, luckyNumbers: [2, 8, 16] };
var dave = { name: "Dave", age: 5, luckyNumbers: [3, 9, 15] };
var kate = { name: "Kate", age: 9, luckyNumbers: [1, 2, 3] };

var friends = [anna, dave, kate];

```

To get information from this array, you use the array index to pick a friend and then the key to get a specific detail. For example:

- To get the second friend (Dave), you'd write: `friends[1]`. This returns the object `{ name: "Dave", age: 5, luckyNumbers: [3, 9, 15] }`.
- To get Dave's name, you'd write: `friends[1].name`. This gives you `"Dave"`.
- To get the second lucky number from Anna's object, you'd write: `friends[0].luckyNumbers[1]`. This gives you `8` because `friends[0]` is Anna's object, `luckyNumbers` is the array `[2, 8, 16]`, and index `1` in that array is `8`.

Think of it like opening a series of boxes: first, you pick the right friend from the array (using the index), then you open their object to find the right key, and sometimes you open another box (like an array) inside that object to get the final piece of information.

3 Exploring Objects in the Chrome Console

When you're coding in JavaScript, the Chrome browser's console is a fantastic tool for exploring arrays and objects. It's like a magnifying glass that lets you peek inside your data.

Let's say you type `friends[1]` into the Chrome console. You'll see something like this:

```
Object { name: "Dave", age: 5, luckyNumbers: Array[3] }
```

Notice the little triangle next to the object? If you click it, the console expands to show all the keys and values, like:

```
name: "Dave"
age: 5
luckyNumbers: Array[3]
```

You can even click on `luckyNumbers: Array[3]` to see the numbers inside: `[3, 9, 15]`. This is super helpful for checking what's inside your arrays and objects, especially when they get complex. It's like unpacking a box to see what's inside, layer by layer.

To try this yourself, open Chrome, press `F12` to open the Developer Tools, go to the Console tab, and type in your array or object. Click the triangles to explore the data.

This is a great way to debug your code and make sure your arrays and objects are set up correctly.

4 Using Objects to Link Strings and Values

Objects are brilliant for linking a string (like a name) to a value (like a number or another piece of data). This is useful when you want to keep track of information in a way that's easy to look up.

For example, imagine you're keeping track of how much money your friends owe you. You could create an object where the keys are your friends' names and the values are the amounts they owe:

```
    var owedMoney = {};
    owedMoney["Anna"] = 3;
    owedMoney["Jimmy"] = 5;

```

Here, `owedMoney` is an object where the key `"Anna"` is linked to the value `3` (dollars), and the key `"Jimmy"` is linked to the value `5`. To check how much Anna owes, you'd write:

```
    owedMoney["Anna"]; // Returns 3

```

This is like a phone book: the name (key) helps you quickly find the number (value). You can add more friends to the object anytime by assigning a new key-value pair, like `owedMoney["Kate"] = 10`.

5 Storing Information About Your Movies

Let's put everything together with a practical example: storing information about a movie collection. You can use an object where each key is a movie title, and each value is another object containing details about that movie. This is called a **nested object**.

Here's an example:

```
  var movies = {
    "Finding Nemo": {
      releaseDate: 2003,
      actors: ["Albert Brooks", "Ellen DeGeneres", "Alexander Gould"],
      format: "DVD"
    },
    "Star Wars: Episode VI - Return of the Jedi": {
      releaseDate: 1983,
      duration: 134,
      actors: ["Mark Hamill", "Harrison Ford", "Carrie Fisher"],
      format: "DVD"
    },
    "Harry Potter and the Goblet of Fire": {
      releaseDate: 2005,
      duration: 157,
      actors: ["Daniel Radcliffe", "Emma Watson", "Rupert Grint"],
      format: "Blu-ray"
    }
  };

```

This `movies` object is like a digital library. Each movie title is a key, and its value is an object with details like the release date, actors (stored in an array), and format (DVD or Blu-ray).

To find information about a movie, you can access it like this:

```
  var findingNemo = movies["Finding Nemo"];
  console.log(findingNemo.format); // Outputs "DVD"
  console.log(findingNemo.actors[1]); // Outputs "Ellen DeGeneres"

```

You can also add a new movie to the collection:

```
  var cars = {
    releaseDate: 2006,
    duration: 117,
    actors: ["Owen Wilson", "Bonnie Hunt", "Paul Newman"],
    format: "Blu-ray"
  };
  movies["Cars"] = cars;

```

To see all the movie titles in your collection, you can use the `Object.keys()` method:

```
Object.keys(movies);
// Outputs: ["Finding Nemo", "Star Wars: Episode VI - Return of the Jedi", "Harry Potter and the Goblet of Fire", "Cars"]

```

This method returns an array of all the keys in the `movies` object, which is handy for listing everything in your collection.

6 Why This Matters

Arrays and objects are powerful because they let you organise data in a way that makes sense. Arrays are great for ordered lists (like a list of friends), while objects are perfect for linking keys to values (like a movie title to its details). When you combine them, you can create complex structures to store all kinds of information, like a dinosaur database, a list of friends, or a movie collection.

Here's a quick comparison:

- **Arrays**: Use numbers (indices) to access items. They're ordered, so the position matters.
- **Objects**: Use strings (keys) to access values. They're not ordered, so the key is what matters.

By nesting arrays and objects, you can build structures that are both flexible and powerful, making your programs more useful.

7 What's Next?

You've now learned how to combine arrays and objects in JavaScript. You can create arrays of objects, access nested values, explore data in the Chrome console, use objects to link strings and values, and build structured collections like a movie database. These skills are the foundation for writing more complex programs.

In future lessons, we'll explore conditionals and loops, which let you add decision-making and repetition to your code. These will make your arrays and objects even more powerful, allowing you to search, filter, and manipulate your data in exciting ways.

For now, try experimenting with your own arrays and objects. Maybe create an array of your favourite games or an object for your book collection. Play around in the Chrome console to see what's inside, and have fun coding!