
SORTABLE MOVIES DATA

Addding Properties to an Array of Objects

Making Arrays of Objects Sortable by key

in ascending / descending order

---

In this lesson we will make a web page of movie info, consisting of a grid of divs, each containing a movie poster image and info about the film.

Let's start by checking out the final product:

1. Open **06.06-Sortable-Movies.html** and preview it in the browser.

The movies load in alphabetical order, although leading articles (A, The) are ignorned, so "A Beautiful Mind" starts with "B" and "The Deer Hunter" starts with "D".

From the **Sort by..** menu:

2. Choose **Year**. The movies are reordered from oldest to newest; check **Descending** to reverse the order (newest to oldest).

3. Choose **Length**. The movies are reordered from shortest to longest; check **Descending** to reverse the order (longest to shortest).

4. Choose **Title**. The movies are reordered alphabetically, by title from A-Z; check **Descending** to reverse the order (Z-A).

The JS code consists of four main parts:

- **movies.js**, the movie dataset, which is an array of objects
- a loop that adds some new properties to the objects
- a **sortMovies** function that sorts the movies by key, per the menu choice
- a **renderMovies** function which makes and outputs the movie info

5. Open **movies.js**. It contains an array, **movies**, of objects, each with **name, year** and **mins** (minutes) properties:

```
const movies = [
    { name: "2001: A Space Odyssey", year: 1968, mins: 149 },
```

```
        { name: "Ali", year: 2001, mins: 157 },
        -- ETC. --
        { name: "The Wizard of Oz", year: 1939, mins: 101 },
    ];
```

6. Switch from **FINAL.js** to **START.js** and reload the page.

The movies are gone. We will now code it all again from scratch.

7. Start by getting the **movie-holder** div, for displaying the movies:

```
    const movieHolder = document.querySelector('.movie-holder');
```

**adding three properties to the movies objects**

- **hrMin**
- We want to display the movie running time in 00:00 format.
- For this we need to convert total minutes to hours:minutes.
- We will add th 00:00 time format to the object as a property, **hrMin**
- **imgFile**
- We need the movie name for concatenating the image file path, but the movie names and file name don't exactly match.
- We will convert the movie names to image file name format.
- We will addd the image file name to the object as a property, **imgFile**.
- **noAThe**
- We will make a property to store the movie name with no leading "A" and "The". This will make it easier to sort by movie name.

We will start by adding the **hrMin** property.

8. Set up a loop to iterate the **movies** array:

```
    for(let i = 0; i < movies.length; i++) {
    }
```

9. Inside the loop, start by simplifying the current movie object:

```
        let movie = movies[i];
```

10. Next, get the **mins** property, and simplifiy it as a variable:

```
        let mins = movie.mins;
```

11. Get the **hours** by dividing **mins** by **60** and rounding down:

```
        let hours = Math.floor(mins / 60);
```

12. Use modulo **%** to get the remainder minutes:

```
        mins = mins % 60;
```

13. Add a leading zero if **mins** is less than 10. This way, we get **2:03**, not **2:3**.

```
        if(mins < 10) mins = '0' + mins;
```

14. Add a key, **hrMin**, to the object. The value is the **hours:mins** format:

```
        movie.hrMin = `${hours}:${mins}`;
```

Now, let's move on to adding the **imgFile** property:

15. Still in the loop, get the name of the movie and make it lowercase:

```
        let imgFile = movie.name.toLowerCase();
```

16. Use **replaceAll()** to replace the spaces in the movie names with hyphens, since the image file names have hyphens instead of spaces between the words:

```
        imgFile = imgFile.replaceAll(' ', '-');
```

17. Remove punctuation from the movie names by replacing commas, apostrophes, colons and periods with nothing.

```
        imgFile = imgFile.replaceAll("'", "");
        imgFile = imgFile.replaceAll(":", "");
```

```
        imgFile = imgFile.replaceAll(",", "");
        imgFile = imgFile.replaceAll(".", "");
```

18. Complete the file name by tacking on '.jpg' at the end:

```
        imgFile += '.jpg';
```

19. Add the **imgFile** property to the object:

```
        movie.imgFile = imgFile;
```

**making a property with no leading articles**

Now for the part where we make a property for the movie name, sans leading "A" or "The". This is the name that we will sort by.

20. Still in the loop, get the movie's first two and first four characters:

```
        let first2chars = movie.name.slice(0, 2);
        let first4chars = movie.name.slice(0, 4);
```

21. If the first two characters are "A ", make a new property **noAThe**, set equal to the full name, starting at the third character (at index 2) and going to the end:

```
        if(first2chars == "A ") {
                movie.noAThe = movie.name.slice(2);
        }
```

22. Else if the first four characters are "The ", make a new property **noAThe**, set equal to the full name, starting at the fifth character (at index 2) and going to the end:

```
        } else if(first4chars == "The ") {
                movie.noAThe = movie.name.slice(4);
        }
```

23. Else, the movie does not start with "A" or "The", so just set **noAThe** to equal the original name:

```
        } else {
                movie.noAThe = movie.name;
        }
```

24. Add one more line before closing the loop: a call to the **renderMovies()** function. This function runs on page load when the loop ends, and also whenever a choice is made to the **Sort by..** menu.

```
        renderMovies();
    } // end loop
```

25. Log the updated **movies** array. It should have all six properties: the original three: name, year and mins as well as the three new ones: hrMin, imgFile and noAThe:

```
    console.log(movies);
```

**making the divs and images dynamically**

Now that the movie objects have their new properties, we are ready to output movies to the DOM. Each movie will be in a div containing an image and some text. The same loop used to make the new properties can also be used to output the movie info.

**renderMovies()** loops through the movies array and makes and outputs the grid of movies:

26. Define the **renderMovies()** function. First thing to do is clear the **movieHolder** of any existing content:

```
        function renderMovies() {
                movieHolder.innerHTML = "";
        }
```

27. Set up a loop to iterate the movies array and simplify the current movie by passing it to a variable:

```
        for(let i = 0; i < movies.length; i++) {
                let movie = movies[i];
        }
```

28. Next in the loop, make a div, give it its class and output it to **movieHolder**:

```
        let divvy = document.createElement('div');
        divvy.className = 'divvy';
        movieHolder.appendChild(divvy);
```

29. Still in the loop, make an image object, set its source to the movie's jpg and output it to the div:

```
        let moviePoster = new Image();
        moviePoster.src = 'images/' + movie.imgFile;
        divvy.appendChild(moviePoster);
```

30. Below the image, output the text info for the movie:

```
        divvy.innerHTML += `${movie.name}<br>${movie.year} -
${movie.hrMin}`;
```

31. Close the loop, close the function and Reload the page. The grid of movies should be back, although the sort feature doesn't work yet. That's next.

**sorting movies**

Now for the user interactivity part: enabling the user to sort the movies by Title (name), Year and Duration (mins). Here's how that works:

- a **change** to the **select** menu calls the **sortMovies** function
- the function gets the selected **value**, which matches an object key
- the function sorts the movie objects by that **key**
- if the **Duration** checkbox is checked, the sort is reversed
- the function ends by calling the **renderMovies** function, which outputs the freshly sorted movies

32. Get the select menu and have it call the **sortMovies** function:

```
        const sortMenu = document.getElementById('sort-menu');
        sortMenu.addEventListener('change', sortMovies);
```

33. Get the checkbox. When a change occurs (check/uncheck), run an inline function that reverses the order of the movies and calls **renderMovies()** to refresh the display:

```
        const descCB = document.getElementById('desc');
        descCB.addEventListener('change', function() {
                movies.reverse();
                renderMovies();
        });
```

In a function, **this** refers to the object that called the function, which in the case of **sortMovies** is the menu itself. When the function is called, **this.value** is the menu choice, which equals the name of the key to sort by.

34. Define the **sortMovies()** function and get the menu choice, which is the sort key:

```

```

```
function sortMovies() {
        let sortKey = this.value;
}
```

In the last lesson, we learned that the sort() method works differently for strings and numbers. Our sort menu choices are:

- Title (name), which is a string
- Year, which is a number
- Duration, which is also a number in minutes
- we will use if-else logic to choose string or number sort.
- for a number sort, **sortKey** can be either **year** or **mins**.

35. If **sortKey == "name"**, do the string sort on the **noAThe** key, which is the name of the movie with no leading "A" or "The":

```
if(sortKey == "name") {
        movies.sort(function(a, b) {
                if(a.noAThe > b.noAThe) {
                        return 1;
                } else {
                        return -1;
                }
        });
}
```

35. Else the key is not "name", so do the number sort. Use the dynamic property access syntax, the square bracket, with **sortKey**, which is either "year" or "mins":

```
} else {
        movies.sort(function(a, b) {
                return a[sortKey] - b[sortKey];
        });
}
```

36. If the Duration checkbox is checked, reverse the sort:

```
if(descCB.checked) movies.reverse();
```

37. Also have the checkbox reverse the order of the movies whenever it is checked or unchecked:

```
descCB.addEventListener('change', function() {
        movies.reverse();
```

```
                renderMovies();
        });
```

38. Still inside sortMovies(), call the renderMovies() function and end the sortMovies() function:

```
           renderMovies();
        } // end sortMovies()
```

39. Reload the page and put the application through its paces. The sort feature should be working.

- **NEXT: Lesson 06.06**
- **Next: Lesson 06.07**