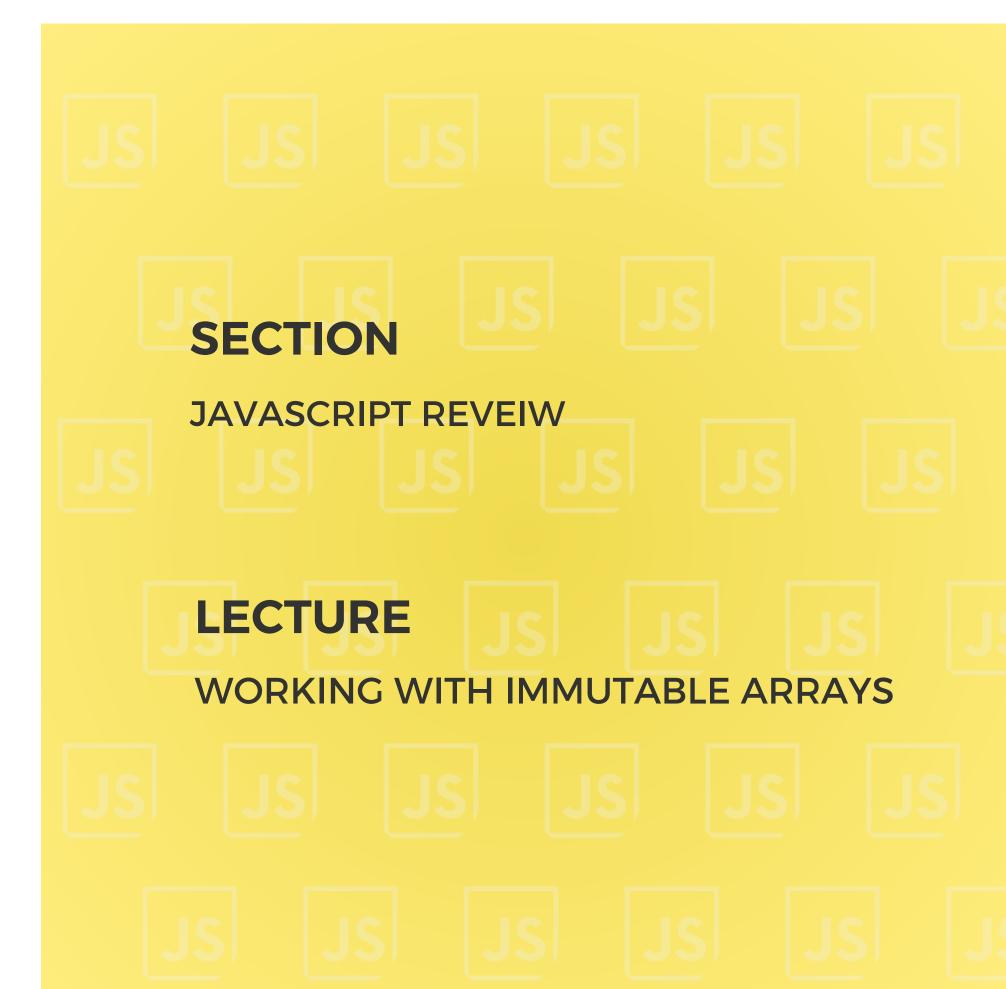


THE COMPLETE FRONT-END DEVELOPMENT





BEFORE WE START

- full In JavaScript, Arrays are a convenient way for storing and manipulating data.
- F However, their **mutability** often leads to unexpected changes and unintended side effects, making debugging a headache.
- Fortunately, We have Immutable Arrays to answer this problem

IMMUTABLE ARRAYS

- The An immutable array is an array that never changes after creation.
- f Modifying elements doesn't alter the original array, but creates a new one instead.
- Immutable arrays ensure **predictable data behavior**, **facilitate easier debugging** by maintaining a clear data history, and enhance performance by **avoiding unnecessary rerenders**.

ADDING AN ELEMENT

- framework in the serving the original array.
- A new array is created with the added element.
- We use the spread operator (...) to create a new array with the added element.

```
const newMovie = {
   id: 6,
   itile: "The Shawshank Redemption",
   publicationDate: "1994-10-14",
   director: "Frank Darabont",
   };
   const moviesAfterAdd = [...data, newMovie];
```

DELETING AN ELEMENT

- f Immutable arrays maintain the original array structure while removing elements.
- A new array is created without the deleted element.
- The wast with the filter method to create a new array excluding the deleted element.

```
const moviesAfterDelete = moviesAfterAdd.filter((movie) => movie.id !== 2);
```

EDITING AN ELEMENT

- framework in the property in the second of t
- A new array is created with the edited element.
- The wast with the map method to create a new array with the edited element.

```
const updatedMovies = movies.map((movie) => {
   return movie.id == 1 ? { ...movie, director: "Rayan" } : movie;
});
```

SEE YOU SOON...