# Destructuring Objects and Arrays

* Useful to get data out of an object or an array.

## Objects

* For objects, we rely on the NAME OF THE PROPERTY
* We use curly brackets {}
* const { title, author } = book2;
* console.log("title", title);
* console.log("author", author);
  + it’s crucial to use the same variable names as in the Object

## Arrays

* for arrays, we rely on the order of the items in the array
* we use square brackets []
* const [primaryGenre, secondaryGenre] = genres;
* console.log("Primary Genre", primaryGenre);
* console.log("Secondary Genre", secondaryGenre);

# Rest Operator – […]

* [ primaryGenre, secondaryGenre, …arrayName]
* This will create a new array from the rest of the array that’s left
* const [primGen, secGen, ...otherGens] = genres;
* console.log("primGen", primGen);
* console.log("secGen", secGen);
* console.log("restGens", otherGens);

# Spread Operator – […] - Arrays

* This is used AFTER the = sign
* It will take all the values from the array and will create a new array that will have also the new value
* const newGenres = [...genres, "epic fantasy"];
* console.log("new Genres", newGenres);

# Spread Operator – […] – Objects

* we can add new properties to an Object
* // Spread Operator - Objects
* const updatedBook = { ...book2, moviePublicationDate: "2001-01-01" };
* console.log("updatedBook", updatedBook);
* we can also overwrite properties in an object
* //  we can also OVERWRITE properties
* const updatedBook2 = {
* ...book2,
* moviePublicationDate: "2022-02-20",
* pages: 1190,
* };

# Template Literals

* Create strings that contain JS variables
* We use ` ` and ${}

# Arrow Functions

# Short Circuiting

* We ca use the logical operators to SHORT CIRCUIT some logic
* SHORT CIRCUITING means that it will return the 1st value without even looking at the 2nd value
* // && operator will automatically return the 1st operator when the 1st opperator is false
* console.log(
* "first operator is true so we get the 2nd operator ----- ",
* true && "some string"
* );
* console.log(
* "first operator is false so we automatically get the first operator ---- ",
* false && "another string"
* );

# OR – Short circuiting

//  when the 1st operator is false, we get the 2nd operator

console.log(true || "some string");

console.log(false || "another string");

* This might return a WRONG VALUE for falsey values ( 0, ‘’, null, undefined)
* That’s why we are now using the ??

# ?? – nullish coalescing operator

* Same as OR || but it works for FALSEY values

# Optional Chaining - ? operator

* We can use the ? operator for values that might be undefined and then we combine them with the ?? (nullish coalescing) to set a default value

# Immutable Arrays

## Add

* We are using the spread Operator to add an element to the array
* NewArr = […array, newElement]

## Delete a book

* We use the FILTER method
* const booksAfterDelete = booksAFterAdd.filter((item) => item.id != 5);

## Update a book

* we use the MAP method
* const booksAfterUpdate = booksAfterDelete.map((item) => {
* // return item.id === 6 ? {...item, item.publicationDate:"THIS IS THE UPDATE"} : item;
* return item.id === 6
* ? { ...item, publicationDate: "THIS IS THE UPDATE" }
* : item;
* });
* console.log("books after update", booksAfterUpdate);

# Async JS Promises

## FETCH API

const data = fetch("https://jsonplaceholder.typicode.com/todos")

  .then((response) => response.json())

  .then((data) => console.log("data", data));

* The fetch API returns a PROMISE
* We use .then to handle the PROMISE response and to transform it into json
* Which returns a new PROMISE
* We use again .then to handle the JSON

# Async / Await Functions

* // Async / Await
* async function getResults() {
* const response = await fetch("https://jsonplaceholder.typicode.com/todos");
* const data = await response.json();
* console.log("new data from the Async/ Await function", data);
* }
* getResults();