ONLINE_LIBRARY

Ayongezwa Ndamase

CONTENT:

JsDoc and Comments

- *I first used @throws to check If the imported variable 'books' is invalid or not(line 14)*
- *Used @type to check the kind of an array the variable matches is, it is an array due to its arrangement in data.js file from it was imported*
- *Page is @type number*
- color themes for day and night modes are @type {Object}
- I *Filtered by title, author, and genre (191)*

SCRIPT AND ELEMENT MODULES

```
import {
  BOOKS_PER_PAGE,
  books,
  genres,
  authors
  } from "./data.js";
```

- I exported 4 variables from file called data.js into script.js(main JS file).
- I linked both the JS files(data and script) to the HTML
- I also used JS to add values to some of the HTML part

VARIABLES & LEXICALS

```
const remaingBooks = () => {
const currentBooks = document.querySelectorAll('preview');
const allSeenBooks = currentBooks.length;
const remaining = books.length - allSeenBooks;
return remaining
}
```

- Variables were used throughout the code, mostly with the use of declarations ‘const’, once or twice, ‘let’;
- Above is an example of an instance of a lexical scope whereby variables are only used within the function they are found in.
- The above code subtracts from the total books everytime the more button is pressed.

CONDITIONAL STATEMENTS, COMPARISONS AND TYPES

```
if (selectedTheme === 'day') {
    setTheme(day);
    cancelButton.disabled = false;
    saveButton.disabled = true;
  } else {
    setTheme(night);
    cancelButton.disabled = true;
    saveButton.disabled = false;
  }
});
```

This piece of code is one of conditionals examples of codes in my project. It states that if our condition is day(or equal to); then the cancel button must be disabled and so on. The '===' is a comparison, true or false are types, which are booleans, there are many other types used in different parts of the code, such as the integers and strings.

BUILT-IN OBJECTS

EG. 1:

```
const fragment = document.createDocumentFragment()

const extracted = books.slice(0, 36)
```

- This example from the code uses built-in object that determines the amount of items to appear at a time. 36 books to be shown on the page initially.

```
} else {
      result = books.filter((book) => {
        return book.title.toLowerCase().includes(title.toLowerCase());
      });
    }
```

- The above line of code also contains a built-in object called .filter() used to narrow down the options for the user. These are among a few that will be covered in the code.

OBJECT LITERALS

```
const day = {
    dark: '10, 10, 20',
    light: '255, 255, 255',
}

const night = {
    dark: '255, 255, 255',
    light: '10, 10, 20',
}
```

The object literal notation makes it easy to create and define the object and its properties all at once.

In the example I provided, ‘day’ and ‘night’

- The properties within them, ‘dark’ and ‘light’ are defines simpler and in a less time consuming manner

DOCUMENT OBJECT MODEL

```
const themeSelector = document.querySelector('[data-settings-theme]');
const cancelButton = document.querySelector('[data-settings-cancel]');
const saveButton = document.querySelector('#save-button');

themeSelector.addEventListener('change', () => {
  const selectedTheme = themeSelector.value;
```

This code snippet fetches the `dataset` , from the HTML file and then changes its function, or gives it function… it does this without directly going to the HTML file, only use of JavaScript.

OBJECT AND ARRAY DESTRUCTURING

```
export const createPreview = (props) => {
  for (const { author, image, title, id } of extracted) {
  const { author: authorId, id, title } = props;
  const element = document.createElement('button');
  element.classList = 'preview';
  element.setAttribute('data-preview', id);
```

This code snippet is an example of object/array destructuring, which is simply the act of pulling or extracting an item from an array/object

- In this instance, the code uses object destructuring to extract the `author` and `title` properties from the `props` object and assigns them to variables `authorId` and `title`,

FUNCTIONS

```
const remaingBooks = () => {
const currentBooks = document.querySelectorAll('preview');
const allSeenBooks = currentBooks.length;
const remaining = books.length - allSeenBooks;
return remaining
    }
```

- This example shows a function, a function is attribute of JavaScript that enables a complex function to be achieved in one or two steps from building up a function and then returning it.
- The example above automates the addition of books on screen, the subtraction from the original total of books .

Events and Listeners

```
const lightToggleBtn = data.home.theme
lightToggleBtn.addEventListener("click", (event) => {
  event.preventDefault();
  lightToggleDialog.showModal();
})
```

- This code defines a constant variable and assigns it the value.
- It attaches an event listener to variable that listens for a "click" event. W
- hen the button is clicked, the code inside the callback function will execute.
- In this case , I intend t5o determine the theme