DS3103 Webutvikling

Candidate 1031

Høyskolen Kristiania

DS3103

28.11.2022

The port is set to 7215 and 5171

Facts learned from Full-stack using React, .NET C# ASP net MVC & SQLite

Service Module

- When using CRUD functions provided by context like delete will return data as empty string, put will return as number.
- All axios GET calls will return in one form or another. For instance, get method by "title" will not return data object itself, but we can retrieve data such as status 200 Ok.
- Get by ID will return data containing the object.
- Get all will also return all data as an array.

Context

- CRUD functions provided by context will not return anything, as they are promise with operations.
- In occasions like editing an object where we have to retrieve one with same ID as input, one cannot use the getById method provided by context, but rather directly access method from GameServiceModule instead. Reason due to the necessity for mapping the data returned from API call storing the state, thereafter initialize a new object with its current state.

State management

- CRUD functions provided by context will not return anything, as they are promise with operations.

- In occasions like editing an object where we have to retrieve one with same ID as input, one cannot use getById method provided by context, but rather directly access method from GameServiceModule instead. Reason due to the necessity for mapping the data returned from API call storing the state, thereafter initialize a new object with its current state.

A side note to state management

- It is also worth to keep in mind that whilst we do have the possibility to use provided props from context, we have to differentiate between "may use" and "has the possibility" to use. I will try to refer to an instance, suppose you want to retrieve one element by property, it is more optimal to handle its own state, rather relying on dependency injection from parent. A separate axios call via service would be more appropriate. The return data would be then passed as an argument to the setter for this component in order to evaluate or render interchangeably.

How to react with React

- React will render only when a state is being changed. We can use setters to set the state of a 'thing' and pass by reference to an event listener, such as "onClick" and "onChange". This action will cause React to react, no pun intended, thus we will get a side effect and reevaluated state.

Pass by reference - Objects/Arrays/Functions

```
// GameList is used as a "passive active" component using dependency
injection from context.
// We can pass by reference in TSX and return as an expression.
    return (
        <>
            <section className="row" >
                    { getGameItems() }
            </section>
        </>
    );
```

The user actions are required to be dispatched from an element like a button or input field.

When clicked or each keystroke, that side effect will kick in.

1. onChange will take immediate effect, commonly inside an input field, listening to user input.
2. onClick will take effect when being clicked on commonly on a button. The button refers to a function by convention, using the name 'on' + 'handler' to invoke the setter.

```
const onClickHandler = async ( ) => {
        if( userInput.current ) {
            const gameWithId = await GameServiceModule.getGameById(
parseInt(userInput.current.value) )
            setChosenGame( gameWithId )
        }
    }
return (
    <input onClick={ onClickHandler } type="button" className='btn
btn-primary' value="Search" />
)
```

The pitfall of using 'row ID' and auto increment

- since auto increment will generate an ID for each record, if we delete any records in between, the gap will never be filled.
- If we emptied the database, the ID will still continue its count from the last record and never 'reset'
- Although this could be fixed with a SQL query in sqlite3.

  How to Reset Autoincrement Number Sequence in SQLite? - Designcise)

```
```sql
UPDATE `sqlite_sequence` SET `seq` = 0 WHERE `name` = 'table_name';
```
> in order to use sqlite3 with javascript we can use npm package
[sqlite3 with
javascript](https://www.sqlitetutorial.net/sqlite-nodejs/connect/)
```

- I did manually execute inside db Browser for the exam to reset ID issue
- I will also address that I set the input limit from 0 - X on purpose, just to visualize on occasions if the last record from the entity has been removed, auto increment will set to previous "position", if you will, hence it will go beyond selection limits, thus the query against database will not be valid, if this was not clear enough.

```
- Id | Name
- 1  | Game 1
- 2  | Game 2 <- delete this
- 3  | Game 3
-
- Id | Name
- 1  | Game 1
- 3  | Game 3 <- then delete this
```

```
-
-  Id | Name
-  1  | Game 1
-  4  | Game 2 <- new added game adapt to last state
-  ```
-  > Apparently there is a Index sheet inside table called
   *sqlite_sequence* where you can manually adjust it back to 0.
```

CSS & Bootstrap

- "App.tsx" is the 'main' container wrapper for main content
- "GameList.tsx" is the 'row
- GameItems.tsx" is the 'col'
- Headers are outside the 'main' and has own sticky position
- Footer is just awkward to React. No comments. What is this invincible blob at bottom when zooming the page out to 30% view, the worst of them all.

- Since we render from root element, if we don't have enough fillers of one page the footer will act weird and stick to its parent leaving a gap between bottom and window.

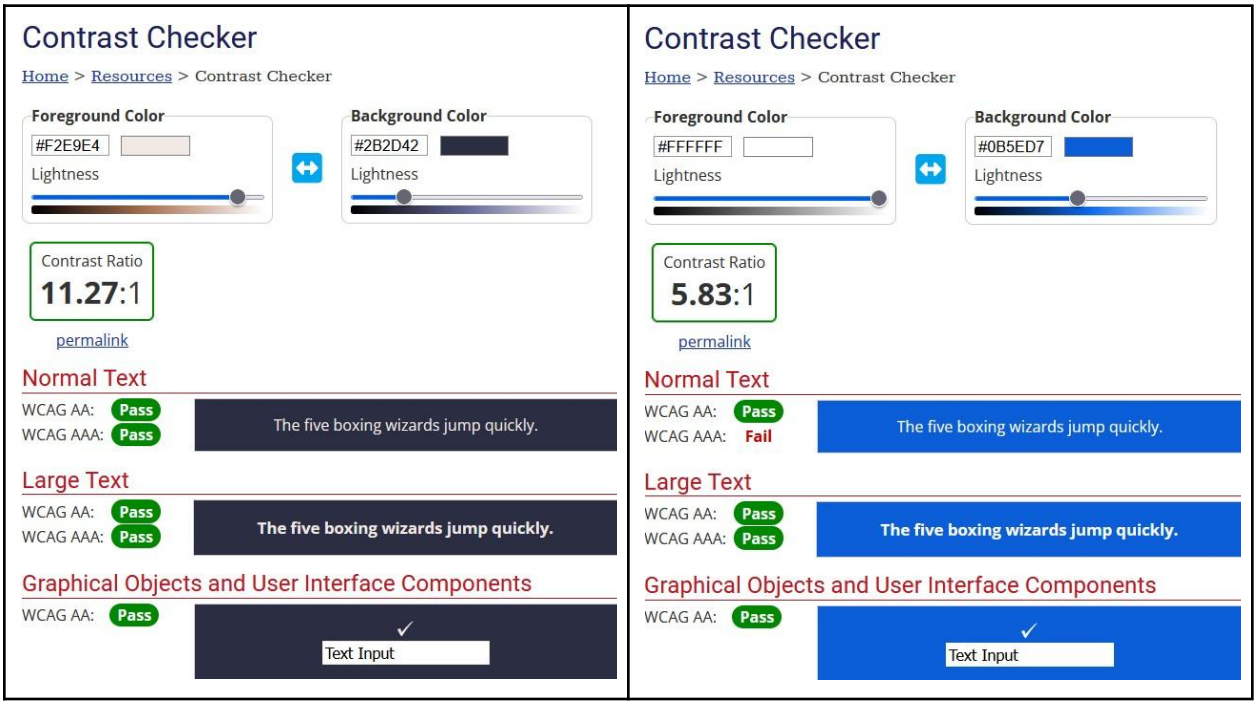
Universal Design

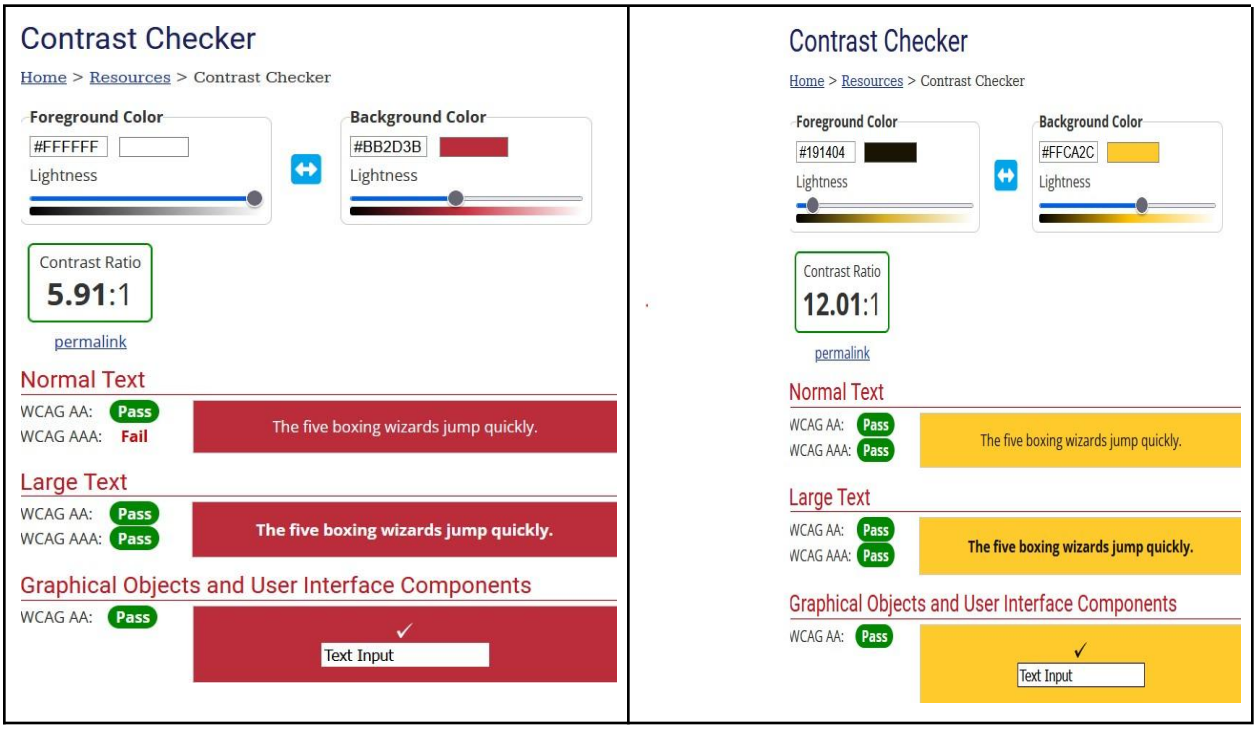
1. Universal Design is about designing with inclusiveness in mind, making a solution for those with disabilities and increasing the accessibility for all. Before making a design, we need to understand the problem, thus we can make a design for that problem. Disabilities can be classified into different forms such as sensory, physical and cognitive.

Contrast Check

2. Contrast is about improving the readability and picking colors that complements each other in order to create a good harmony. While being selective with colors, we should also have an alternative solution for people with sensory disabilities such as color blindness. According to the World Wide Web Consortium(W3C) the luminosity ratio should be at least AA.

3.





4. <img alt=””> Images should include an alternative text in case the visual information is not available. When an equivalent text is provided, people with disabilities have the accessibility to that information, such as transcript for deaf people and vice versa. It is worth to note that some search engines cannot see images.
5. Semantic coding plays a role in Universal Design. To reduce cognitive overload, we may refine or code structurally and clean code. It should be intuitive to read and by using semantic HTML correctly, it will also be included in search engines. We may adapt to specific methodology, such as BEM to follow an HTML structure. The most important is using semantic tags has built-in accessibility features.

“Internet IS for everyone – but it won’t be unless WE make it so.”

—Vint Cerf