INFT-22001 WotC Code Documentation

Jeffrey:
In the ReactJS website, the main features I worked on were the overall base site, navbar, and the pagination of the site.
The ReactJS website has 4 screens, About, Contact, Landing, and Products.jsx, this is a change from the template which had only one.
The navbar navigates between the pages using HTML links.
App.js defines the routes to each page, providing the path that is used by the top nav bar. Routes are from the extra package react-router-dom.

What shows on each page is defined in the .jsx file in src/screens.
Each page has its own content as well as the footer.

The Products page has the most content and would contain several set icons up top, followed by a list of current and future products, it would then contain showcase cards from those sets, followed by a plug for a featured set product and standard pricing for the product types (Draft, Set, and Collector booster packs).

About would have recent blog entries talking about new products, in-person events, and format news such as rules changes, clicking on an article generates an event which would bring you to the article if the site had a backend, more articles would find older articles if the site had a back end.

Contact has multiple forms for the user to type in and submit feedback, this would send their report to the administrators who could view it from the backend.

Chris:

Products page:

HTML Structure:

1. **Header Section (<header>):**
    - Contains the title of the webpage ("MAGIC THE GATHERING: PAST VS PRESENT").
2. **Main Content ( ):**
    - Divided into two sections (<section class="left-side"> and <section class="right-side">).
    - **Left Side (<section class="left-side">):**
        - Displays information about past events/products.
    - **Right Side (<section class="right-side">):**
        - Displays information about upcoming events/products.
        - Similar structure to the left side with an unordered list and list items for each product/event, along with associated images.

CSS Styling (styles-3.css):

1. **Header Styling (header):**
    - Background color set to dark gray (#333).
    - Text color set to white.
    - Padding, text alignment, and font size applied for styling.
2. **Title Styling (h1):**
    - Font size set to 3.3rem.
3. **Body Styling (body):**
    - Background image applied with cover, centered position, and no repeat.
    - This sets the background for the entire body of the page.
4. **Container Styling (.container):**
    - Flexbox styling for the main content container.
    - Justified content with space between items.
    - Padding, border, and box-sizing for layout.
5. **Left and Right Side Styling (.left-side and .right-side):**
    - Background colors and font sizes applied for styling.
6. **Heading 2 Styling (h2):**
    - Text color and font size applied for styling.
7. **Paragraph Styling (p):**
    - Text color applied for styling.
8. **Unordered List and List Item Styling (ul, li):**
    - Font size applied for styling.

JavaScript File (script-3.js):

1. **Audio Control:**
    - Creates a new Audio object with background music file ("background-music-2.mp3").
    - Sets the audio to loop.
    - Defines functions (playAudio and pauseAudio) for playing and pausing audio.
    - Event listeners to play and pause audio.

JIM – LOGIN PAGE AND CONTACT PAGE

LOGIN PAGE:

index.html:

This is the home page with a login button and an introduction text. All elements of the page is contained within a div element with the class container for easy styling.

styles.css:

On the first line is the body selector, which chooses the entire area of the website. Within the body class we have margin and padding = 0 to get make sure the content reachers every edge of the viewport. justify-content and align-items to center to make sure the cross and main axis of the child elements is in dead center of the page. the 100vh in the height is to make sure the body reaches the viewport height, taking up the entire screen. The linear gradient is to start the top color with a shade of blue and slowly transitions to purple at the bottom. the .container class has the text-align center to make sure its elements is centered properly. the font-size: 5rem on the .welcome-text class is to make sure the text is 5 times bigger. About the button, a padding of 10px height and 20px height is added and filled blue with a 30px border radius to give it a rounded edge. rounded-button:hover triggers an effect when you hovers your mouse over the button it zooms up a bit for a bit of dynamic.

script.js:

We use letpasswordAttempts = 0 to represents the initial state then document.getElementById("login-button").addEventListener("click") to make sure the button responds to the click. A while loop is for prompting the user continuously for a password until the correct password ("magicthegathering") is entered or the user cancels the prompt. If the user cancels the prompt (if password === null), the loop is broken. If the user enters a blank password if (password.trim() === ""), an alert is displayed asking them to enter a password. If the user enters an incorrect password, the passwordAttempts variable is incremented by one (passwordAttempts++). If the user makes 5 incorrect attempts, a hint is displayed using the displayHint() function, and the "LOGIN" button is moved down using the moveLoginButton() function. If login successful the content of the webpage is cleared with document.body.innerHTML = "";. A countdown timer and its style is set with CSS. A countdown timer function updateTimer() is defined to update the timer and perform actions when the countdown reaches zero. The countdown timer is then started with updateTimer()

CONTACT PAGE:

index.html:

We have all the elements in a form class, with input type for every field so that when you type anythine without the @ in email it will require you to fill out the field.

script.js:

We use letpasswordAttempts = 0 to represents the initial state then document.getElementById("login-button").addEventListener("click") to make sure the button responds. Then add a submit function to the button that activates when clicked. Then an empty object as the form for user to enter the data in. If every field is filled up, then the alert function will prompt the user with a login success message