

BBC Homepage Clone - Project Documentation

1. Project Overview

This project is a high-fidelity, responsive clone of the BBC homepage, built using vanilla HTML, CSS, and JavaScript. It serves as a fully functional prototype featuring dynamic content filtering, interactive article expansion, real-time search, and a secure user authentication flow.

2. Key Features

- **Dynamic Filtering:** Users can filter news by categories (Technology, Science, Sport, Ethiopia, etc.) using either the sticky header or the comprehensive footer navigation.
- **Immersive Article Viewer:** Clicking any news card opens the story in a full-screen modal overlay, creating a seamless, single-page application (SPA) feel without reloading the page.
- **Real-time Search:** A robust search bar filters articles instantly by matching titles, summaries, and category tags.
- **Secure Auth Flow:** Includes professionally styled Sign-in and Register pages with robust client-side validation.
 - **Secure Passwords:** Enforces complex passwords (min 8 chars, uppercase, lowercase, number, and special character) using Regex.
 - **Email Validation:** Ensures valid email formats during registration.
 - **Interactive Feedback:** Visual error states and descriptive messages guide users through the authentication process.
- **Fully Responsive:** The layout fluidly adapts from a 4-column desktop grid to a single-column mobile view using CSS Grid and Flexbox.
- **Ethiopian Spotlight:** Features a dedicated section for positive news stories related to Ethiopia.

3. File Structure

- `bbc.html`: The core landing page containing the header, news grid, and footer.
- `signin.html`: A clean login portal with integrated credential validation.
- `register.html`: A registration page featuring advanced regex-based form validation.
- `index.css`: Central stylesheet managing layouts, animations, and responsive breakpoints.
- `scripts.js`: The application's "brain," handling filtering logic, interactive states, and date generation for the homepage.
- `*.jpg / *.png`: Optimized assets for article thumbnails.

4. Technical Implementation

CSS Architecture

- **CSS Variables:** At the root of the stylesheet, variables like `--bbc-red` and `--bbc-black` are defined. This allows for global, one-line changes to the site's color scheme, ensuring brand consistency and making maintenance effortless.

- **Hybrid Layout:** The page intelligently combines CSS Flexbox and Grid. Flexbox is used for one-dimensional alignment in the header and footer, perfectly arranging items in a single row. CSS Grid is used for the complex, two-dimensional news gallery, allowing articles to span multiple columns and rows gracefully.
- **Modal Viewer Styling:** The full-screen article viewer is an excellent example of CSS positioning. The container uses `position: fixed` to break out of the normal document flow and overlay the entire viewport. A high `z-index` ensures it sits above all other content. Its visibility is toggled by changing its `opacity` from 0 to 1, with a `transition` applied to create a smooth, professional fade effect.
- **Validation Styling:** User feedback during form validation is handled by a simple but effective `.input-error` class. When validation fails, this class is added to the input field, applying a `2px solid red` border to instantly draw the user's attention to the field that requires correction.

JavaScript Logic

- **Unified Filter Engine:** The `filterArticles` function is the heart of the homepage's interactivity. It simultaneously checks two things on every user interaction: the `currentCategory` selected from the navigation and the `searchTerm` from the search bar. An article is only displayed if it matches both the active category (or if "Home" is selected) and the search query, creating a powerful, intersectional filter.
- **Dynamic Modal Implementation:** The SPA-like article viewer is achieved through efficient DOM manipulation. When a news card is clicked:
 1. The script captures the `src` of the image and the `textContent` of the title and full story from the clicked card.
 2. It then populates the corresponding elements inside the single, hidden `#article-viewer` container.
 3. Finally, it adds an `.viewer-active` class to the viewer to trigger the CSS fade-in `opacity` transition and a `.body-no-scroll` class to the `<body>` to prevent background scrolling. A listener on the "close" button simply reverses these class changes.
- **Event-Driven Form Validation:** The sign-in and registration forms provide instant, client-side feedback without a page refresh. An event listener on the submit button prevents the default form submission and instead runs a validation sequence. It tests the input values against predefined rules (e.g., a secure password regex) and populates an `errors` array with user-friendly messages for any failures. If the array is empty, the action succeeds; otherwise, the errors are displayed.
- **Modular Isolation:** To prevent errors, script logic is carefully isolated. `scripts.js` is built exclusively for the homepage and its specific DOM elements (like the news grid). The validation logic for `signin.html` and `register.html` is kept within those files as inline scripts. This prevents the homepage script from running on pages where its target elements don't exist, which would otherwise cause crashes.

5. Teacher Q&A (22 Key Technical Points)

Question: Why did you use the `<article>` tag for news stories? Answer: It provides semantic meaning, indicating each story is an independent, self-contained piece of content.

Question: What is the benefit of using CSS Variables like `--bbc-red`? Answer: They allow for global consistency and make it easy to update colors site-wide from one location.

Question: How is the news grid layout created? Answer: It uses CSS Grid with display: grid and grid-template-columns: repeat(4, 1fr).

Question: How do you make the first article larger than the others? Answer: By applying grid-column: span 2 to the .hero-card class.

Question: What does box-sizing: border-box do? Answer: It ensures that padding and borders are included in the element's total width and height.

Question: How does the full-screen article viewer work without loading a new page? Answer: It uses a hidden `div` styled with `position: fixed`. JavaScript dynamically copies the article's content into this `div` and makes it visible by changing its `opacity`, simulating a new page.

Question: What is the purpose of data-category attributes? Answer: They act as identifiers that JavaScript uses to match navigation links with specific news articles.

Question: How does the search function filter content? Answer: It iterates through articles and uses `.includes()` to check if the search term exists in the content.

Question: Why is DOMContentLoaded used in the script? Answer: It ensures the JavaScript only runs after the HTML structure is fully loaded and accessible.

Question: How does the mobile "hamburger" menu work? Answer: JavaScript toggles an `.open` class on the navigation, changing its display from none to block.

Question: What does `e.preventDefault()` do in the navigation handler? Answer: It stops the browser from following the link, allowing JavaScript to handle the filtering.

Question: How is the layout made responsive for mobile phones? Answer: A media query at 500px changes the grid to `grid-template-columns: 1fr` for a single-column view.

Question: Why use `object-fit: cover` for article images? Answer: It ensures images fill their containers without being stretched or distorted.

Question: How do the footer links sync with the header navigation? Answer: A `setActiveNav` function updates the `currentCategory` variable and visual active state for all nav links simultaneously.

Question: What is "Event Delegation" in this project? Answer: Attaching one click listener to the parent `<nav>` instead of individual listeners to every link.

Question: How does the "sticky" header work? Answer: The CSS property `position: sticky` with `top: 0` keeps the header at the top while scrolling.

Question: What is the difference between Flexbox and Grid here? Answer: Flexbox is used for 1D alignment in the header; Grid is used for the 2D news gallery.

Question: How are articles hidden during filtering? Answer: A `.hidden` class containing `display: none !important` is added to the elements via JavaScript.

Question: Why did you create a separate register.html? Answer: To demonstrate a complete user flow and maintain consistent styling across multiple pages.

Question: How do the category filter and search bar work together? Answer: The filterArticles function checks both the active category and the search input simultaneously.

Question: Why did you implement password validation using Regular Expressions? Answer: To ensure a high level of security by enforcing specific character requirements (uppercase, numbers, symbols) that are difficult to guess.

Question: How do you handle form validation errors visually? Answer: By toggling an error message container and applying a red border to the specific input field using an .input-error CSS class.