**Project Requirements:**

1. This is a full-page, full-width, full-height component, and needs to work on all screen sizes.
2. When the user clicks this button, an alert appears which reads "Hello, World"
3. Header & Footer to have a size of 100px.
4. Fluid, responsive height for the space between the header and footer
5. Add any image which will cover this entire white space not set via CSS
6. Optimize the image for different screen sizes
7. Add a translucent overlay above the image
8. Add a title that will be in the center of this space, above the image and overlay
9. Add a button under the title, which reads "Click here"
10. Technologies Used: HTML, CSS, JS

**Code Documentation:**

**HTML Code**

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Full Page Component</title>

<link rel="stylesheet" href="assets/css/styles.css">

</head>

<body>

<header>

<h1>Travel the World with Us!</h1>

</header>

<main>

<div class="full-page">

<img src="assets/img/travel\_img.jpg" alt="Background Image">

<div class="overlay"></div>

<div class="content">

<h2>Travelopia</h2>

<button id="clickButton">Click here</button>

</div>

</div>

</main>

<footer>

<p>&copy; 2024 Front-end Project</p>

</footer>

<script src="assets/js/script.js"></script>

</body>

</html>

**Explanation:**

1. **Document Type Declaration (<!DOCTYPE html>)**:
   * Declares the document type and version of HTML being used.
2. **Language Attribute (<html lang="en">)**:
   * Specifies the language of the document for screen readers and search engines.
3. **Meta Tags**:
   * **Charset Meta Tag**: Defines the character encoding for the document (UTF-8).
   * **Viewport Meta Tag**: Ensures proper rendering and scaling on different devices.
4. **Title Tag (<title>Full Page Component</title>)**:
   * Sets the title of the webpage, which appears in the browser tab.
5. **Link Tag (<link rel="stylesheet" href="styles.css">)**:
   * Links the HTML file to an external CSS file (**styles.css**) for styling.
6. **Header Section (<header>...</header>)**:
   * Contains the header content, including the main title of the page.
7. **Main Section (<main>...</main>)**:
   * Contains the main content of the page.
   * Includes a full-page component with an image, overlay, title, and button.
8. **Full-Page Component (<div class="full-page">...</div>)**:
   * Represents the full-page component that occupies the entire viewport.
   * Contains an image, overlay, and content elements.
9. **Image Tag (<img src="your-image.jpg" alt="Background Image">)**:
   * Displays the background image for the full-page component.
   * The **alt** attribute provides alternative text for screen readers and in case the image fails to load.
10. **Overlay (<div class="overlay"></div>)**:
    * Represents a translucent overlay placed above the image.
    * Provides a visual effect to enhance the readability of the content.
11. **Content Section (<div class="content">...</div>)**:
    * Contains the title and button elements positioned over the image.
    * Centered vertically and horizontally within the full-page component.
12. **Footer Section (<footer>...</footer>)**:
    * Contains footer content, such as copyright information.
13. **Script Tag (<script src="script.js"></script>)**:
    * Links the HTML file to an external JavaScript file (**script.js**) for scripting functionality.

**Decisions Taken:**

* **Semantic HTML**: Used semantic HTML elements (**header**, **main**, **footer**) to improve accessibility and maintainability.
* **External CSS and JavaScript Files**: Separated CSS styles and JavaScript functionality into external files for better organization and reusability.
* **Responsive Design**: Ensured the page layout and components are responsive across various screen sizes and devices.
* **Translucent Overlay**: Implemented a translucent overlay to improve the contrast between the background image and content elements.
* **Button Click Functionality**: Added JavaScript functionality to handle button click events, displaying an alert message when the button is clicked.
* **Alternative Text for Images**: Provided descriptive **alt** text for images to enhance accessibility and support users with disabilities.
* **Viewport Meta Tag**: Included a viewport meta tag to ensure proper rendering and scaling on mobile devices.

**CSS Code:**

**styles.css**

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body, html {

height: 100%;

}

header, footer {

height: 100px;

background-color: #333;

color: #fff;

padding: 10px;

text-align: center;

}

main {

height: calc(100% - 200px); /\* Subtract header and footer height \*/

display: flex;

justify-content: center;

align-items: center;

}

.full-page {

position: relative;

width: 100%;

height: 100%;

overflow: hidden;

}

.full-page img {

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

object-fit: cover;

}

.overlay {

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, 0.4); /\* Translucent overlay \*/

}

.content {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

text-align: center;

color: white;

}

button {

padding: 10px 20px;

background-color: #0056a0;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 16px;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #0056b3;

}

**Explanation:**

1. **Universal Selector (\* { ... })**:
   * Resets default margin, padding, and box-sizing for all elements to ensure consistent styling.
2. **Body and HTML (body, html { ... })**:
   * Sets the height of the body and HTML elements to 100% to fill the viewport height.
3. **Header and Footer Styling (header, footer { ... })**:
   * Defines the common styling for header and footer elements, including height, background color, padding, and text color.
4. **Main Section Styling (main { ... })**:
   * Sets the height of the main section to fill the remaining viewport height after considering the header and footer heights.
   * Uses flexbox to center its child elements vertically and horizontally.
5. **Full-Page Component Styling (full-page { ... })**:
   * Positions the full-page component relative to the viewport.
   * Sets its width and height to 100% to cover the entire viewport.
   * Uses **overflow: hidden** to prevent overflow of child elements.
6. **Background Image (full-page img { ... })**:
   * Positions the background image absolutely within the full-page component.
   * Sets its width and height to cover the entire space while maintaining aspect ratio using **object-fit: cover**.
7. **Translucent Overlay (overlay { ... })**:
   * Positions the translucent overlay absolutely above the background image.
   * Uses **rgba(0, 0, 0, 0.5)** to set the background color with 50% opacity, creating a translucent effect.
8. **Content Positioning (content { ... })**:
   * Positions the content (title and button) in the center of the full-page component using absolute positioning and CSS transforms.
   * Centers the text horizontally and vertically within the content container.
9. **Button Styling (button { ... })**:
   * Defines the styling for the button element, including padding, background color, text color, border, border radius, cursor, font size, and transition effect.
10. **Button Hover Effect (button:hover { ... })**:
    * Changes the background color of the button on hover to provide visual feedback to the user.

**Decisions Taken:**

* **Resetting Default Styles**: Resetting default margin, padding, and box-sizing ensures consistent styling across different browsers.
* **Flexible Layout**: Using flexbox (**display: flex**) in the main section allows for easy vertical and horizontal centering of content.
* **Translucent Overlay**: Adding a translucent overlay enhances the visibility of content over the background image.
* **Responsive Design**: Using percentage-based dimensions and flexible positioning ensures the component adapts to different screen sizes.
* **Button Styling**: Styling the button with padding, background color, and hover effect improves its visual appearance and user experience.
* **CSS Transforms**: Using CSS transforms for content positioning simplifies centering content within the full-page component.

**JS Code:**

**script.js**

document.addEventListener('DOMContentLoaded', function() {

var clickButton = document.getElementById('clickButton');

clickButton.addEventListener('click', function() {

alert('Hello, World');

});

});

**Explanation:**

1. **DOMContentLoaded Event Listener**:
   * Listens for the **DOMContentLoaded** event, which fires when the initial HTML document has been completely loaded and parsed.
2. **Callback Function**:
   * Defines a callback function that runs when the **DOMContentLoaded** event is triggered.
3. **Button Element Selection**:
   * Uses **document.getElementById('clickButton')** to select the button element with the ID **clickButton**.
4. **Button Click Event Listener**:
   * Adds a click event listener to the selected button element.
5. **Alert Message**:
   * Displays an alert message with the text "Hello, World" when the button is clicked.

**Decisions Taken:**

* **DOMContentLoaded Event**: Using the **DOMContentLoaded** event ensures that the JavaScript code executes after the HTML document has been fully loaded and parsed, preventing potential issues with accessing elements that have not yet been created.
* **Separation of Concerns**: Keeping the JavaScript code separate from the HTML and CSS files promotes clean and maintainable code. It adheres to the principle of separation of concerns, making it easier to manage and debug.
* **Event Delegation**: By attaching the click event listener to the button element directly, the code efficiently handles user interactions without relying on inline event handlers or repetitive code.
* **Use of Vanilla JavaScript**: The code demonstrates the use of native JavaScript methods and APIs without relying on external libraries or frameworks. This decision promotes lightweight and efficient front-end development.

**Test Cases:**

1. **Load Page Successfully:**

* Verify that the page loads without any errors.
* Ensure that all elements including header, main content, and footer are visible.

1. **Header and Footer Size:**

* Check that the header and footer have a height of 100 pixels each.
* Ensure that the header and footer have a background color of #333 and white text color.

1. **Full-Page Component Structure:**

* Verify that the full-page component contains an image, translucent overlay, title, and button.
* Ensure that all elements are positioned correctly within the full-page component.

1. **Image Loading:**

* Confirm that the image loads successfully and covers the entire background space.
* Verify that the image is properly optimized for different screen sizes.

1. **Translucent Overlay:**

* Check that the translucent overlay is displayed above the image.
* Ensure that the overlay has the correct opacity level.

1. **Title Positioning:**

* Verify that the title is centered horizontally and vertically within the full-page component.
* Ensure that the title font color is white and the font size is appropriate.

1. **Button Functionality:**

* Click the "Click here" button and verify that it triggers an alert.
* Confirm that the alert message displays "Hello, World".

1. **Responsiveness:**

* Resize the browser window and verify that the full-page component adapts to different screen sizes.
* Ensure that all elements remain visible and properly positioned on various devices and screen resolutions.

1. **Performance:**

* Measure the page load speed using browser developer tools or performance testing tools.
* Ensure that the page loads within an acceptable time frame, considering the size of the image and other resources.

1. **Cross-Browser Testing:**

* Test the project in different web browsers such as Chrome, Firefox, Safari, and Edge.
* Ensure consistent rendering and functionality across different browsers.

-Submitted By **Meenadshi M**