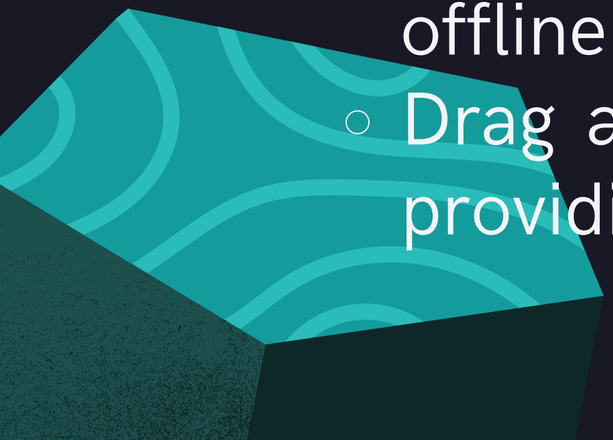
Native Support for APIs

- HTML5 provides built-in APIs to enhance user interaction and functionality:
    - Geolocation API: Accesses the user’s geographical location, enabling location-based services.
    - Web Storage API: Offers a way to store data locally in the user’s browser (Local Storage and Session Storage) for offline use and improved performance.
    - Drag and Drop API: Enables drag-and-drop functionality, providing a more interactive user experience.

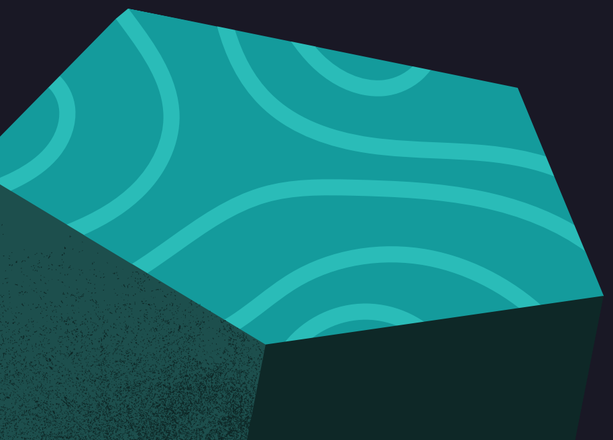


Introduction to Web Accessibility

Definition of Web Accessibility

- Web accessibility is the practice of designing and developing websites that can be used by people with disabilities, including visual, auditory, motor, and cognitive impairments.

- It emphasizes ensuring equal access to information and functionality for all users.



Importance of Making Websites Accessible

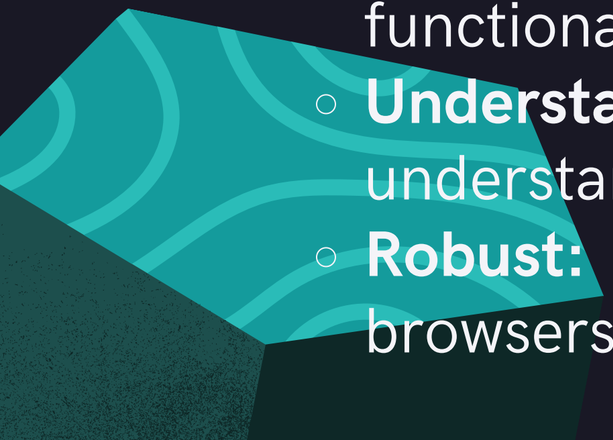
- **Legal Requirements:** Many countries have laws and regulations mandating accessible web design (e.g., ADA in the United States, WCAG compliance).
- **Wider Audience Reach:** An accessible website can reach a larger audience, including people with disabilities and older adults.
- **Improved Usability for Everyone:** Accessibility features often enhance overall usability for all users, regardless of ability. For example, text alternatives can also benefit users in low-bandwidth situations.

Overview of Web Content Accessibility Guidelines (WCAG)

- WCAG provides a set of guidelines to help developers create accessible web content.

- The guidelines are organized into four key principles:

  - **Perceivable:** Users must be able to perceive the information presented (e.g., providing text alternatives for non-text content).
  - **Operable:** User interface components must be operable (e.g., ensuring all functionality is available from a keyboard).
  - **Understandable:** Information and operation of the user interface must be understandable (e.g., using clear and simple language).
  - **Robust:** Content must be robust enough to work across different browsers and assistive technologies.



Key Principles of Accessibility

- **Perceivable**
  - Information must be presented in a way that users can perceive.

  - **Strategies:**
    - Provide text alternatives for non-text content (e.g., images, audio, and videos) using alt attributes.
      - Example: <img src="image.jpg" alt="Description of image">
    - Use sufficient color contrast to ensure readability (e.g., contrast between text and background).
    - Provide captions and transcripts for audio and video content.
    - Ensure that content can be resized up to 200% without loss of functionality.

Key Principles of Accessibility (cont')

- **Operable**
  - User interface components must be operable by all users.
  - **Strategies:**
    - Make all interactive elements usable via a keyboard without requiring mouse interaction (e.g., navigation links, buttons).
    - Provide clear navigation options and structure (e.g., breadcrumb trails, logical tab order).
    - Implement time limits carefully, allowing users to extend or disable timeouts as necessary.
    - Ensure that users can easily find and operate tools such as search boxes and forms.

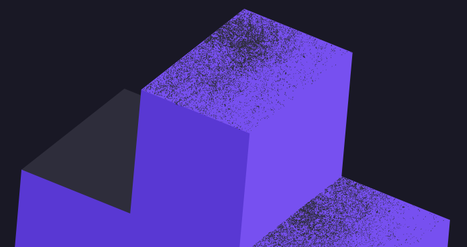
Key Principles of Accessibility (cont')

- **Understandable**
  - Information and operation of the user interface must be clear and understandable.
  - **Strategies:**
    - Use consistent and predictable navigation and interface elements throughout the site.
    - Provide clear instructions and error messages when users interact with forms.
      - Example: Highlight errors in red and provide suggestions for correction.
    - Write in plain language whenever possible, avoiding jargon and complex vocabulary.
    - Use headings, lists, and other elements to organize content logically and improve readability.

Key Principles of Accessibility (cont')

- **Robust**
  - Content must be robust enough to be reliably interpreted by a wide variety of user agents, including assistive technologies.
  - **Strategies:**
    - Use valid HTML and ARIA (Accessible Rich Internet Applications) roles to convey meaning to assistive technologies.
    - Ensure compatibility with current and future user agents (browsers, screen readers).
    - Regularly test your website with various browsers and tools to verify accessibility.
    - Follow best practices for coding, ensuring clean and maintainable code.

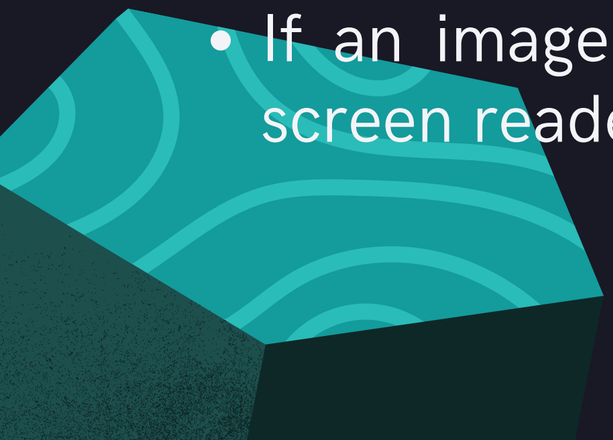
Accessible HTML Practices



Use of Alt Attributes for Images

- **Importance:** Alt attributes provide important context for images, ensuring users who cannot see them still receive the information.

- **Guidelines:**
  - Use descriptive alt text that conveys the purpose of the image.
  - Bad example: alt="image1"
  - Good example: alt="A family having a picnic in a park"
-
- If an image is purely decorative, use an empty alt attribute (alt="") so screen readers can skip it.



Proper Heading Structure

- Properly structured headings help users navigate content easily.

- **Guidelines:**
  - Use headings (<h1>, <h2>, <h3>, etc.) in a hierarchical manner to structure content.
  - Ensure that only one <h1> is used per page, typically as the main title.
  - Use <h2> for major sections and <h3> for subsections, maintaining a logical structure.

ARIA Roles and Properties

- Accessible Rich Internet Applications (ARIA) enhance the accessibility of dynamic web content.

- Usage:
  - Use ARIA roles to describe elements and their functionality:
    - Example: Click me!
  - Apply ARIA properties to convey additional context:
    - Example: Use aria-label to provide a hidden label for interactive elements.
  - Be cautious when using ARIA; it should supplement, not replace, semantic HTML.

Keyboard Navigation

- Importance: Ensures that all users, including those who cannot use a mouse, can navigate the site effectively.

- Guidelines:
  - Ensure that all interactive elements (links, buttons, forms) are accessible via keyboard shortcuts.
  - Use the tabindex attribute to manage custom focus order when necessary.
  - Provide visual focus indicators (like outlines) for keyboard users to show which element is currently selected.

Implementing Accessibility in Forms