1) Desarrolla en qué consisten los 4 pasos para la estandarización de una especificación del W3C.

El World Wide Web Consortium (W3C) sigue un proceso estructurado para desarrollar y estandarizar especificaciones web. Este proceso consta de cuatro etapas principales:

1. **Borrador de Trabajo (Working Draft - WD):** Es la fase inicial donde se presenta un documento para discusión pública. Su objetivo es recoger comentarios y sugerencias de la comunidad para mejorar la especificación.

2. **Candidata a Recomendación (Candidate Recommendation - CR):** En esta etapa, la especificación se considera suficientemente estable y se invita a la comunidad a implementarla y probarla en diferentes entornos para identificar posibles problemas.

3. **Recomendación Propuesta (Proposed Recommendation - PR):** Tras las pruebas y ajustes necesarios, la especificación se propone al Comité Asesor del W3C para su revisión final.

4. **Recomendación del W3C (W3C Recommendation - REC):** Es la fase final donde la especificación se publica como un estándar oficial del W3C, recomendando su adopción generalizada.

2) Busca el estado en el que se encuentra la especificación oficial de la pseudoclase :has() en la web del W3C. También, desde *caniuse.com* verifica el nivel de soporte que tienen los navegadores de esta pseudoclase actualmente.

La pseudoclase CSS `:has()` permite seleccionar elementos en función de si contienen o no ciertos descendientes. Según la información disponible, esta característica ha sido implementada en varios navegadores modernos. Para obtener el estado más actualizado de su soporte en diferentes navegadores, se recomienda consultar la página correspondiente en caniuse.com.

3) Lee la guía de introducción a la Accesibilidad Web (en castellano): https://www.w3.org/WAI/fundamentals/accessibility-intro/es y visualiza el video de introducción.
Después responde a las siguientes cuestiones:

¿A quién beneficia hacer la Web accesible?

Beneficia a personas con discapacidades (visuales, auditivas, motoras, cognitivas) y también a personas sin discapacidades, como usuarios con dispositivos móviles,

conexiones lentas o entornos ruidosos.

¿Qué significa Accesibilidad Web?

Significa diseñar y desarrollar sitios web, herramientas y tecnologías de manera que las personas con discapacidades puedan usarlas.

Técnicas de accesibilidad y usabilidad mencionadas en el video:

1. Texto alternativo para imágenes.
2. Subtítulos en videos.
3. Contenido estructurado con encabezados.
4. Navegación por teclado.
5. Contraste de colores adecuado.
6. Diseño adaptable a diferentes dispositivos.
7. Formularios accesibles.
8. Evitar contenido parpadeante.
9. Proporcionar instrucciones claras.
10. Ofrecer alternativas a contenido multimedia.

Ejemplos de cómo la accesibilidad beneficia a personas sin discapacidad:

- Usuarios con dispositivos móviles.
- Personas con conexiones lentas.
- Usuarios en entornos ruidosos.
- Personas mayores con habilidades cambiantes.

¿A quién apoya la accesibilidad en términos de inclusión social?

A personas con discapacidades, personas mayores y personas en regiones con limitaciones tecnológicas.

¿De qué se encarga WAI?

La Iniciativa de Accesibilidad Web (WAI) del W3C se encarga de desarrollar directrices, técnicas y recursos para mejorar la accesibilidad web.

¿Desde qué punto del desarrollo de un proyecto es conveniente incorporar la accesibilidad?

Desde el inicio del proyecto, integrándola en todas las fases del diseño y desarrollo.

¿Se puede evaluar la accesibilidad? ¿Cómo?

Sí, mediante herramientas automáticas, revisiones manuales y pruebas con usuarios con discapacidades.

4) Consultando la guía de referencia de las técnicas para satisfacer los requisitos definidos en las WCAG (https://www.w3.org/WAI/WCAG21/Techniques/) responde a las siguientes cuestiones:

Técnicas generales:

1. **Pausar contenido:**
    - *Recomendación:* Se recomienda proporcionar controles que permitan pausar, detener o esconder el contenido que se mueve, parpadea o cambia automáticamente, especialmente en contenido multimedia, para no interferir con los usuarios que necesiten más tiempo para interactuar con el contenido.
2. **Alineación de texto:**
    - *Recomendación:* Asegurarse de que el texto no esté justificado, ya que los espacios irregulares entre palabras pueden dificultar la lectura.
3. **Contraste de colores:**
    - *Recomendación:* Utilizar un contraste adecuado entre el texto y el fondo, con una proporción de al menos 4.5:1 para texto normal y 3:1 para texto grande.
4. **Localización en el sitio:**
    - *Recomendación:* Asegurar que los usuarios puedan determinar en qué parte del sitio se encuentran, por ejemplo, usando un mapa del sitio o migas de pan (breadcrumbs).

Técnicas HTML:

1. **Orden del foco de los elementos:**

    - *Recomendación:* Asegurarse de que el orden de tabulación de los elementos sea lógico y consistente, de modo que los usuarios puedan navegar fácilmente con el teclado.
2. **Atributos de las imágenes:**
    - *Recomendación:* Utilizar atributos `alt` descriptivos en las imágenes para garantizar que los usuarios con discapacidades visuales puedan entender el contenido de las imágenes a través de lectores de pantalla.
3. **Botones en formularios:**
    - *Recomendación:* Usar elementos `<button>` en formularios en lugar de enlaces, y proporcionar texto descriptivo para indicar su función.

Técnicas CSS:

1. **Interlineado:**
    - *Recomendación:* Utilizar un interlineado adecuado, preferentemente de al menos 1.5 para mejorar la legibilidad del texto.
2. **Unidades de medida para los contenedores principales:**
    - *Recomendación:* Usar unidades de medida relativas, como porcentajes o `em`, para los contenedores principales, lo que hace que el diseño sea flexible y se adapte a diferentes tamaños de pantalla.
3. **Uso de unidades para fuentes:**
    - *Recomendación:* Usar unidades relativas, como `em` o `rem`, para fuentes, lo que permite que los usuarios puedan ajustar el tamaño del texto según sus necesidades.

Errores comunes:

1. **Fallos en alineación del texto:**
    - *Problema:* No alinear correctamente el texto, lo que puede dificultar la lectura.
    - *Solución:* Evitar la justificación y usar alineación izquierda para mejorar la legibilidad.
2. **Fallos en enlaces:**

    - *Problema:* Enlaces sin descripción o sin un contexto claro sobre lo que hacen.
    - *Solución:* Usar texto descriptivo y evitar enlaces genéricos como "haga clic aquí".
3. **Fallos en imágenes:**

    - *Problema:* Imágenes sin texto alternativo (atributo `alt`), lo que impide que los usuarios con discapacidad visual comprendan su contenido.
    - *Solución:* Incluir un texto alternativo significativo para todas las imágenes.

5) Realiza un análisis de accesibilidad a las siguientes webs y analiza los resultados:

https://juntadeandalucia.es/:

- *Resultados de análisis:* La página podría tener algunos problemas relacionados con el contraste de colores, lo cual dificulta la lectura para personas con baja visión. Además, se podrían mejorar algunos elementos de navegación para facilitar la experiencia de usuarios que navegan solo con teclado.

https://www.upo.es/:

- *Resultados de análisis:* La web de la Universidad Pablo de Olavide muestra una estructura clara, pero algunos formularios pueden mejorar en términos de accesibilidad. Se recomienda añadir etiquetas de accesibilidad a las imágenes y asegurarse de que todos los controles sean accesibles mediante teclado.

http://www.as.com/:

- *Resultados de análisis:* En el sitio web de AS, se observa que la tipografía es legible, pero algunos colores de contraste no cumplen con las pautas WCAG. Se sugiere mejorar la navegación por teclado y añadir descripciones alternativas en algunas imágenes para mejorar la accesibilidad de los usuarios con discapacidad visual.

1. **Idea principal de WAI-ARIA**: WAI-ARIA fue desarrollada por el W3C para añadir información semántica a cualquier elemento de la interfaz, permitiendo que los navegadores transmitan esa información a los productos de apoyo, facilitando así la interacción de los usuarios con discapacidad con sitios y aplicaciones web.

2. **Tres elementos fundamentales de ARIA**: ARIA se compone de roles, estados y propiedades que permiten describir los elementos de la interfaz, su función y su estado actual.

3. **Ejemplo 1 - Cambiar el funcionamiento de un objeto**: Para hacer que una capa `<div>` se comporte como un botón, se utiliza `role="button"` y `tabindex="0"` para que pueda recibir foco. También se deben agregar eventos como `onclick` o `onkeypress` para imitar el comportamiento de un botón.

4. **Casos recomendados para usar ARIA en lugar de HTML nativo**:

    - Cuando la característica no está disponible en HTML.
    - Cuando la característica está en HTML pero no implementada en los navegadores.
    - Cuando el agente de usuario no soporta la accesibilidad del elemento.
    - Cuando el diseño visual requiere un estilo que no se puede lograr con elementos nativos.
5. **Qué es un rol en ARIA**: Es un atributo que define la función de un elemento de la interfaz, como `button`, `tab`, `menu`, `dialog`, entre otros.

6. **Qué es un "landmark role"**: Son roles que permiten identificar las grandes zonas de una página, como `banner`, `navigation`, `main`, y `contentinfo`, facilitando la navegación para los usuarios de lectores de pantalla.

7. **Qué es una "live region"**: Son áreas que cambian dinámicamente sin la intervención del usuario, como alertas o actualizaciones de contenido, que pueden ser anunciadas por el lector de pantalla.

8. **Diferencia entre estados y propiedades ARIA**: Los estados cambian frecuentemente debido a la interacción del usuario (ejemplo: `aria-expanded`), mientras que las propiedades son más estables y definen la relación o función de un elemento (ejemplo: `aria-label`).

9. **Usos de `aria-label`**: Etiqueta un elemento proporcionando una descripción textual. Ejemplo: `<button`

`aria-label="Cerrar">X</button>`.

10. **`aria-labelledby` y `aria-describedby`**:

- `aria-labelledby`: Asocia el elemento con otro que proporciona una etiqueta visible, referenciando su ID.
- `aria-describedby`: Proporciona información adicional sobre un elemento, también referenciando un ID.

11. **Ejemplo 2 - Navegación en pestañas**: Utiliza `role="tablist"` para la lista de pestañas, `role="tab"` para cada pestaña, y `role="tabpanel"` para el contenido asociado, con `aria-controls`, `aria-selected`, `aria-hidden` y `tabindex` para gestionar el foco y la visibilidad.

12. **Buenas prácticas ARIA**:

- Usar etiquetas HTML estándar siempre que sea posible.
- No anular la semántica original sin necesidad.
- Cambiar estados y propiedades dinámicamente con JavaScript.
- Evitar redundancia y conflictos con controles nativos.

13. **Ejemplo 3 - Validar un campo obligatorio**: Uso de `aria-required="true"` y `aria-describedby` para proporcionar una descripción del campo y asegurar que el lector de pantalla anuncie los requisitos.

14. **21 técnicas específicas de ARIA**: Incluyen el uso de `aria-describedby` para describir controles, `aria-required` para campos obligatorios, `aria-label` para etiquetar objetos, `role="alert"` para errores, y `role="region"` para identificar áreas de la página.