[](https://www.google.com.mx/url?sa=i&url=https%3A%2F%2Fwww.uaa.mx%2Fportal%2Fnuestra-universidad%2Finstitucion%2Flogotipo%2F&psig=AOvVaw3AptWZ9ucL-HKODPyNeugi&ust=1591112800952000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCPD5tKT74OkCFQAAAAAdAAAAABAD)

9-6-2020

Maestra: Georgina Salazar Partida

Materia: Tecnologías WEB

Integrantes:

* González Campos Carlos Alejandro
* Guerrero Carrera Jessica
* Ramírez González Isaías Emmanuel
* Robledo Aguilar José Ramón
* Serna Dávila Mariela Teresa

Accesibilidad web (lectores de pantalla)

**Accesibilidad web (lectores de pantalla)**

* Un documento donde expliquen el proceso de investigación, opciones que analizaron y lo que finalmente los llevo a seleccionar una Librería, incluir explicación de librería y proceso de instalación.
* Un proyecto Angular que permita la lectura de alguna sección de la vista, incluir icono de accesibilidad que a partir del clic inicie la lectura y tener manera de detenerla.
* Un link a una videoconferencia donde le expliquen a sus compañeros el código y como fue el proceso de investigación.

Para poder comenzar con esta investigación, consulté que son los lectores de pantalla son un software que permite la utilización del sistema operativo y las distintas aplicaciones mediante el empleo de un sintetizador de voz que “lee y explica” lo que se visualiza en la pantalla, lo que supone una ayuda para las personas con graves problemas de visión.

EL funcionamiento de los lectores de pantalla se puede representar mejor con el siguiente diagrama:



**ARIA**

ARIA es buena para unir áreas con problemas de accesibilidad que no se pueden controlar con HTML nativo. Trabaja permitiéndote especificar atributos que modifican la forma en que un elemento se traduce al árbol de accesibilidad. Por ejemplo:

En el siguiente fragmento, usamos un artículo de lista como un tipo de casilla.



A pesar de que esto funciona bien para usuarios con visión, un lector de pantalla no brinda indicaciones de que el elemento sirve como casilla, asi que, los usuarios de baja visión pueden perderse el elemento por completo.

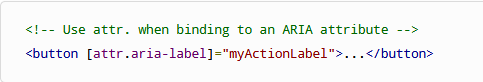
Con el uso de atributos ARIA podemos brindarle al elemento la información que falta para que el lector de pantalla pueda interpretarla correctamente. Aquí se agregaron los atributos role y aria-checked para identificar explícitamente el elemento como casilla y para especificar que está marcada de manera predeterminada. Este artículo de lista ahora se agregará al árbol de accesibilidad y un lector de pantalla los informará correctamente como casilla.



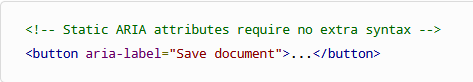
**Atributos de accesibilidad**

Construir una experiencia web accesible a menudo implica establecer **atributos ARIA** para proporcionar un significado semántico donde de otro modo podría faltar. Utilice la sintaxis de la plantilla **de enlace de atributos (binding)** para controlar los valores de los atributos relacionados con la accesibilidad.

Cuando se vincula a atributos ARIA en Angular, debe usar attr. prefijo, ya que la especificación ARIA depende de los atributos HTML en lugar de las propiedades de los elementos DOM.



Tenga en cuenta que esta sintaxis solo es necesaria para los enlaces de atributos (binding). Los atributos ARIA estáticos no requieren sintaxis adicional.



**Componentes UI de Angular**

La biblioteca de **material Angular**, la cual es mantenida por el equipo de Angular, es un conjunto de componentes de interface de usuario (UI) reutilizables cuyo objeto es ser totalmente accesible. El **Kit de desarrollo de componentes (CDK)** incluye el paquete a11y que proporciona herramientas para admitir diversas áreas de accesibilidad. Por ejemplo:

* LiveAnnouncer se utiliza para anunciar mensajes para usuarios de lectores de pantalla que utilizan una región aria-live.
* La directiva cdkTrapFocus atrapa el enfoque de la tecla Tab dentro de un elemento. Úselo para crear una experiencia accesible para componentes como los cuadros de diálogo modales, donde el enfoque debe ser limitado.

Argumentando elementos nativos

Los elementos HTML nativos capturan una serie de patrones de interacción estándar que son importantes para la accesibilidad. Al crear componentes Angular, debe reutilizar estos elementos nativos directamente cuando sea posible, en lugar de volver a implementar comportamientos bien soportados.

Por ejemplo, en lugar de crear un elemento personalizado para una nueva variedad de botones, puede crear un componente que use un selector de atributos con un elemento <button> nativo. Esto se aplica más comúnmente a <button> y <a>, pero se puede usar muchos otros tipos de elementos.

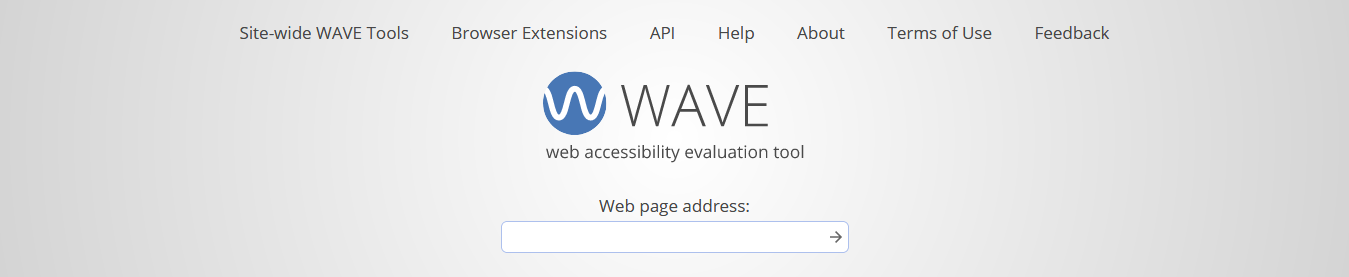
Utilizando contenedores para elementos nativos

A veces, usar el elemento nativo apropiado requiere un elemento contenedor. Por ejemplo, el elemento <input> nativo no puede tener hijos, por lo que cualquier componente de entrada de texto personalizado debe envolver un <input> con elementos adicionales. Si bien puede incluir el <input> en la plantilla de su componente personalizado, esto hace imposible que los usuarios del componente establezcan propiedades y atributos para el elemento de entrada, En su lugar, puede crear un componente contenedor que use proyección de contenido para incluir el control nativo en la API del componente.

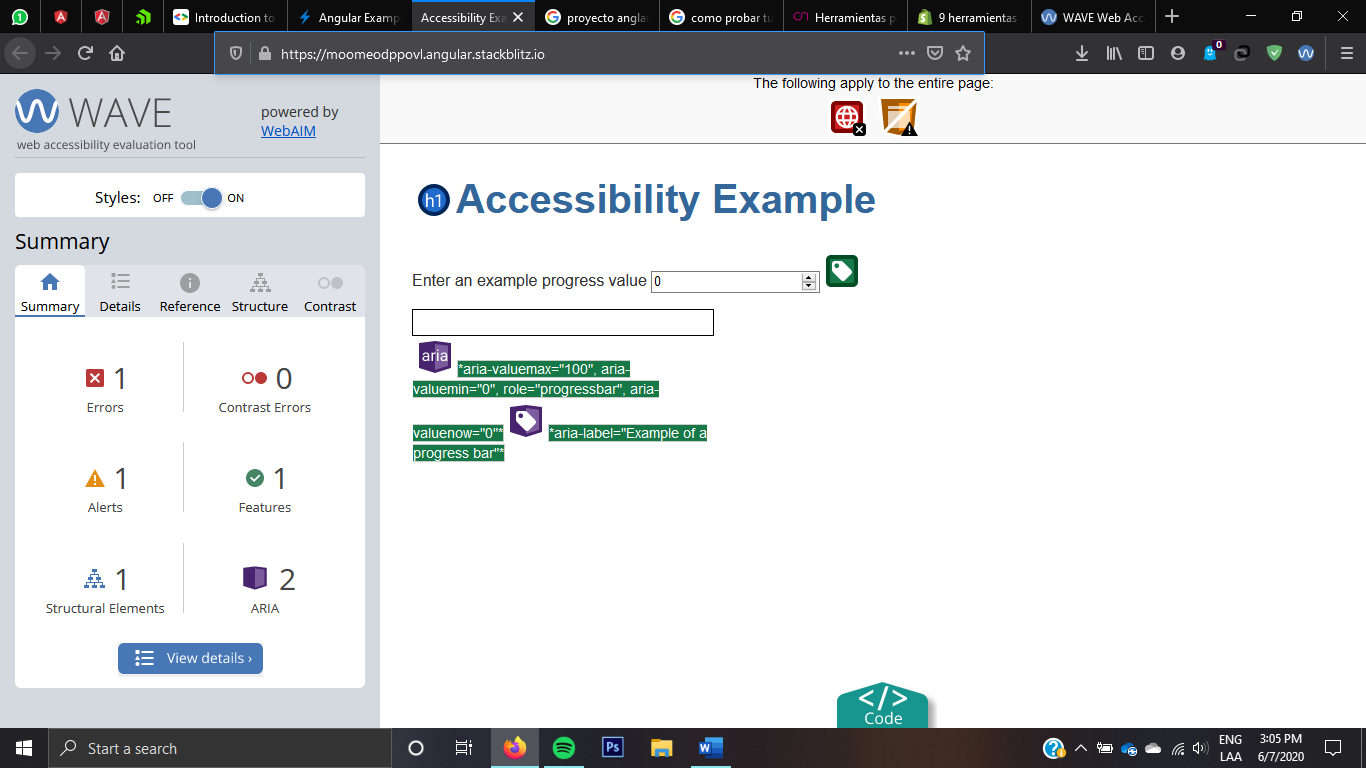
**¿Cómo probar la accesibilidad?**

Al momento de investigar cuales son los lectores de pantalla más populares para probar sitios web me crucé con uno que se llama WAVE, Wave es una herramienta gratuita de comunidad diseñada para facilitar la revisión de la accesibilidad web. Las pruebas evaluarán varios problemas de accesibilidad, que incluye contraste de colores a los atributos ARIA y se basan en las normas de cumplimiento de WCAG.

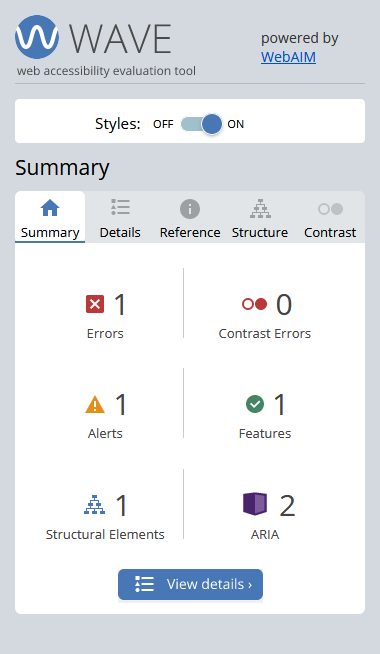
WAVE se puede utilizar mediante su sitio web <https://wave.webaim.org/> el cual cuenta con un buscador en donde puedes insertar una URL de algún sitio y lo abrirá en esa misma pestaña de manera en que puedes evaluar la accesibilidad.



De otra manera, se puede realizar una instalación de extensiones mediante los navegadores FireFox y Chrome, para poder utilizar WAVE, simplemente presionaremos el logotipo que se encuentra en la sig. Imagen y podemos comenzar la evaluación.



WAVE cuenta con un menú en el cual se despliegan un resumen de lo que se encontró en la página, como errores, errores de contraste, alertas, características, elementos estructurales y ARIA.

* Errores: Problemas que pueden impactar a ciertos usuarios o fallas para cumplir con WCAG.
* Errores de contraste: Texto que no cumple con los requerimientos de contraste WCAG.
* Alertas: Elementos que pueden causar problemas.
* Características: Elementos que pueden mejorar la accesibilidad si son implementados correctamente.
* Elementos estructurales: Muestra como está estructurado la página.
* ARIA: Señala los lugares en donde se utilizó ARIA. Presenta información con accesibilidad.

Con lo anterior, creo que es una muy buena opción para poder probar la accesibilidad de nuestro sitio.

**¿Cuál librería puedo usar para probar el TTS (text to speach)?**  
En base a la investigación que realicé, encontré una librería que se llama “speak-tts”. El cual permite la lectura de nuestro proyecto sin necesidad de tener instalado algún software.

Para su instalación pondremos en la línea de comandos lo siguiente

npm install speak-tts

Importamos la librería

import Speech from 'speak-tts' // es6

// var Speech = require('speak-tts') //if you use es5

Probamos el soporte de nuestro navegador

const speech = new Speech() // will throw an exception if not browser supported

if(speech.hasBrowserSupport()) { // returns a boolean

console.log("speech synthesis supported")

}

Iniciamos el componente del habla:

const speech = new Speech()

speech.init().then((data) => {

// The "data" object contains the list of available voices and the voice synthesis params

console.log("Speech is ready, voices are available", data)

}).catch(e => {

console.error("An error occured while initializing : ", e)

})

Un ejemplo podría ser lo siguiente:

// Example with full conf

Speech.init({

'volume': 1,

'lang': 'en-GB',

'rate': 1,

'pitch': 1,

'voice':'Google UK English Male',

'splitSentences': true,

'listeners': {

'onvoiceschanged': (voices) => {

console.log("Event voiceschanged", voices)

}

}

})

Para poder leer un texto se hace

speech.speak({

text: 'Hello, how are you today ?',

}).then(() => {

console.log("Success !")

}).catch(e => {

console.error("An error occurred :", e)

})

Link con más propiedades y funciones:

<https://www.npmjs.com/package/speak-tts>