EJERCICIOS SOBRE WEBCOMPONENTS

Para los ejercicios con módulos, tienes un zip denominado "Servidor para enunciado" en el que dispones de los archivos necesarios para arrancar un servidor en Node.js y realizar las pruebas necesarias.

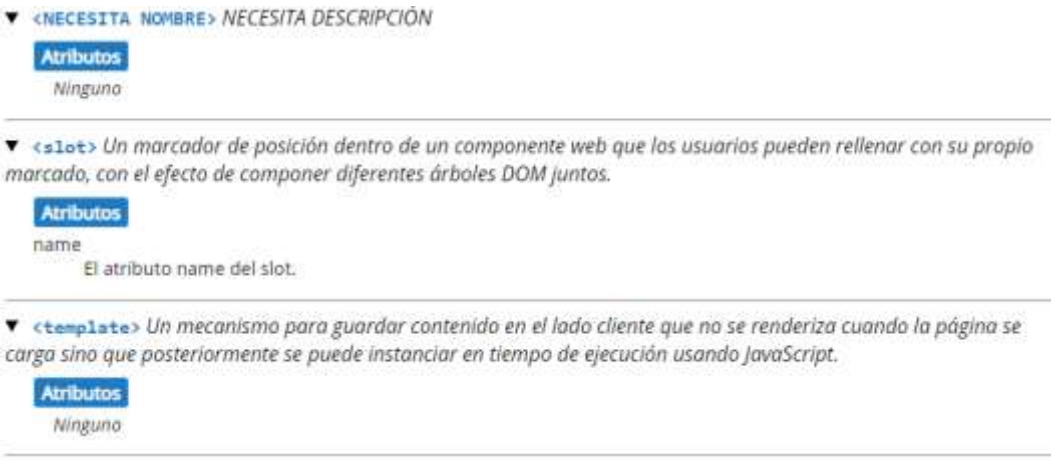
1. Crea un WebComponent que herede de **HTMLElement** con las especificaciones que se indican a continuación
   - El componente se definirá con el nombre **wc-blink**
   - El componente puede recibir tres atributos (los tres son opcionales):
     - ✓ **baseColor:** Un color en un formato válido en CSS, ya sea RGB o hexadecimal. Si este atributo no existe, entonces el valor será "inherit".
     - ✓ **alternativeColor:** Lo mismo que el anterior. Lo ideal es que sea un color diferente a baseColor. Si este atributo no existe, entonces el valor será "transparent".
     - ✓ **changeInterval:** Un número entero que se expresa en segundos. Si no se especifica nada, el componente web considerará 1 segundo.
   - El componente se encarga de cambiar de color el texto de la etiqueta **wc-blink** cada "n" segundos según el valor de **changeInterval** siguiendo la secuencia siguiente: el color inicial es **baseColor**, a los "n" segundos según **changeInterval** cambia a **alternativeColor,** cuando pase este tiempo vuelve a cambiar a **baseColor** y así mientras el navegador siga abierto.

2. Ahora realiza lo mismo que en el ejercicio anterior, pero heredando de **HTMLSpanElement** cambiando la forma de definir la etiqueta, si fuese necesario.

3. Crea un WebComponent con las especificaciones que se indican a continuación.
   - Se trata de un Custom Element basado en la etiqueta **detail** de HTML. A continuación, se recuerda que un detail contiene una etiqueta denominada **summary** y un texto inicialmente oculto definido con otra etiqueta. Cuando se hace click en el texto asociado con summary, se muestra el texto oculto. Un ejemplo sería el siguiente, en el que al hacer click en el texto "Epcot Center" se despliega el párrafo de debajo:

```
<details>
    <summary>Epcot Center</summary>
    <p>Epcot is a theme park at Walt Disney World Resort featuring
       Exciting attractions, international pavilions, award-winning
       fireworks and seasonal special events.
    </p>
</details>
```

   - El objetivo es crear un detail personalizado con el nombre de etiqueta **element-details.** A continuación se muestra una imagen con tres ejemplos de funcionamiento ya desplegados (si hacemos click en la flecha de la izquierda se mostraría solamente la primera línea con el título en azul). Si echas un vistazo, especialmente al primer ejemplo, verás que hay tres etiquetas personalizables.
     - ✓ **<NECESITA NOMBRE>**
     - ✓ **NECESITA DESCRIPCIÓN**
     - ✓ **Ninguno (dentro de atributos)**



- En los otros dos ejemplos alguna de estas etiquetas tiene un valor. Debes configurar el WebComponent para que aparezca por defecto el texto del primer ejemplo, pero que a la hora de definir la etiqueta sea personalizable y se pueda cambiar por el texto que decida el diseñador.
- Los estilos deben ser lo más parecidos posible a la imagen de arriba. El componente debe crearse empleando un template que se carga en el shadow DOM.

4. Crea un WebComponent mediante un módulo de JavaScript con las especificaciones que se indican a continuación:

- El componente se denomina **x-product** y tendrá un aspecto similar al siguiente:

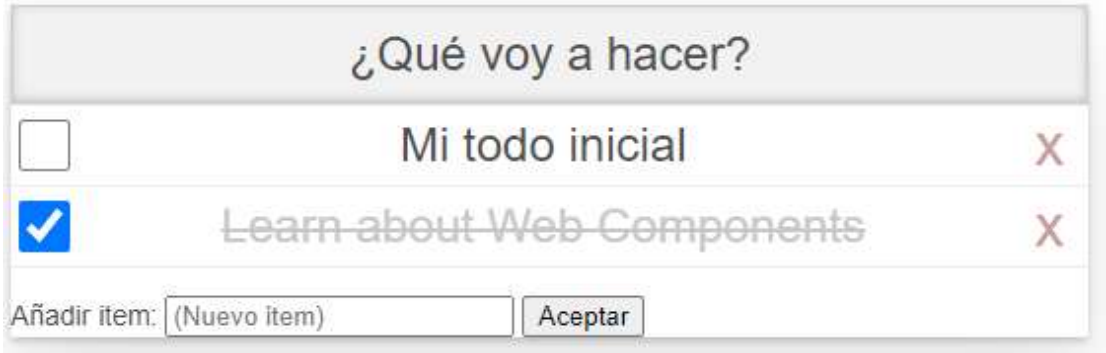

Ruby

- Como puedes observar, es una imagen con un enlace a la derecha que recibe dos atributos:
  - ✓ **data-name:** Es el nombre del enlace que aparece a la derecha.
  - ✓ **data-url:** Es una URL a la que se redirige haciendo click tanto en la imagen como en el texto. Puede ser cualquier URL de prueba. Por ejemplo: https://example.com

- La imagen se carga del servidor de la URL https://s3-us-west-2.amazonaws.com/s.cdpn.io/4621/. Para ello, el componente debe estar preparado para añadir el nombre del atributo **data-name** en minúsculas a la URL añadiendo el formato png. Por ejemplo, si data-name tiene el valor Ruby, entonces la URL será https://s3-us-west-2.amazonaws.com/s.cdpn.io/4621/ruby.png. Las imágenes del componente se reducirán a 150x150.
- El servidor ahora mismo solo contiene tres imágenes de prueba asociadas al atributo **data-name** con valores Ruby, JavaScript y Python.
- Para generar la URL de la imagen se creará un módulo aparte que se importará en el WebComponent. El WebComponent también se creará como un módulo que exporta la clase correspondiente.
- Debes realizar dos versiones: una donde el propio JavaScript que crea el WebComponent lo define y otra que defina el componente el propio HTML importando el módulo correspondiente.

5. Vamos a crear nuestro **select personalizado** que carga todos los países mediante un fichero denominado "countries.json" que puedes descargar del aula virtual en la misma sección que el enunciado. Para ello, tendrás que seguir las siguientes especificaciones:
   - Un JSON se puede cargar con el método fetch como se explica en el siguiente enlace https://www.todojs.com/api-fetch-el-nuevo-estandar-que-permite-hacer-llamadas-http/
   - Una vez cargado, en el método then correspondiente debes crear las diferentes etiquetas option con el atributo value con la clave del objeto y el HTML será el propio valor asociado a la clave.
   - Si la carga del JSON falla, en lugar de dar un error, debes proporcionar un objeto alternativo con algún país de prueba. Por ejemplo: `{"AD": "Andorra", "AE": "United Arab Emirates"}`
   - Lo más sencillo es que extiendas la clase de HTMLSelectElement y crear el componente como un módulo de JavaScript.

6. Uno de los ejemplos más típicos de componente web es una **lista de tareas**, por tanto, vamos a crear una similar a la imagen de debajo con las especificaciones que se indican a continuación.

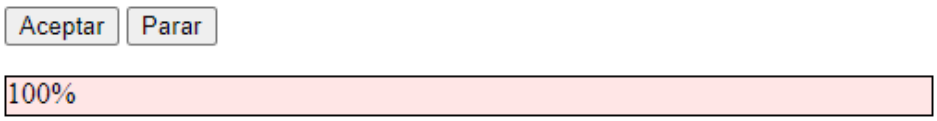


- Cada tarea tiene dos botones:
  - ✓ Un checkbox a la izquierda que se encarga de mostrar si está completada o no. Fíjate en los estilos de la descripción cuando está seleccionado y cuando está en blanco.
  - ✓ Un botón de eliminar a la derecha bastante descriptivo que se encarga de eliminar la tarea.
- Además, puedes añadir nuevos ítems en el formulario de debajo. Se introduce la descripción en el campo de texto "(Nuevo ítem)" y al hacer click en "Aceptar" se añade.
- El componente debe denominarse **my-todo** y para incluirlo en el HTML basta con incluir las etiquetas correspondientes sin parámetros ni texto **<my-todo></my-todo>**.
- Se puede inicializar con alguna tarea, o que aparezca vacío y se añadan con el formulario correspondiente.
- También es posible dividir el componente en varios. Por ejemplo, crear un componente aparte para cada ítem y luego incluirlo en el componente my-todo final.

7. Crea un WebComponent denominado **fancy-tabs**, que consiste en un panel con pestañas similar a la imagen de debajo y con las especificaciones que se describen más adelante:

- El componente consiste en:
  - ✓ Un número n de pestañas y un número n de elementos de contenido. Para garantizar el funcionamiento, el número de pestañas y elementos de contenido debe ser el mismo.
  - ✓ Cada pestaña puede ser una etiqueta diferente de HTML. Se recomienda h1, h2, button o a.
  - ✓ De la misma forma, cada elemento de contenido se puede reemplazar por cualquier etiqueta, aunque es más recomendable un elemento en bloque como div o section.
- El componente debe estar definido claramente para que el HTML se asocie correctamente. Por ejemplo, si hay tres pestañas como en la imagen, se pueden definir primero y luego el contenido asociado a cada una.
- Al hacer click en alguna pestaña, se cambia a la actual y se muestra el contenido asociado con dicha pestaña.
- A la hora de definir el componente, puedes preparar el componente para que haya alguna pestaña concreta seleccionada por defecto. Por ejemplo, a partir de algún atributo denominado selected.

8. Vamos a combinar los conocimientos adquiridos de animaciones y WebComponents. Para ello, vamos a crear un componente llamado **progress-bar** similar a la imagen de debajo.



- La barra de progreso está inicialmente en blanco sin mostrar nada. La captura de pantalla de arriba indica el estado cuando ha terminado.
- La barra recibe un atributo denominado **seconds**, que es la cantidad que tarda en completarse. Si este campo no existe o es menor que cero, entonces se mostrará una alerta.
- El usuario hace click en el botón "Aceptar" para comenzar a completar la barra de progreso. Además de rellenar el contenido con un color de fondo, se debe indicar el porcentaje actual en cada instante como se puede observar.
- El usuario puede parar la barra de progreso en el botón correspondiente denominado "Parar". La barra de progreso no se puede reanudar y al hacer click en "Aceptar" se comenzaría desde cero.
- Si se detecta un cambio por JavaScript del atributo **seconds**, la barra de progreso volverá al valor 0. Puedes añadir un botón adicional en la página para probar esta funcionalidad.