Creación del proyecto

## La creación de una página web para el LaPAB (Laboratorio de Laboratorio de Procesos Acoplados Biofísicos) de la Universidad Austral de Chile (UACH) surge a partir del taller sobre Olas de Calor Marinas celebrado en Concepción el 9 de abril, un evento donde fue posible entrar en contacto con los investigadores del laboratorio y surgió este proyecto de colaboración conjunta para favorecer la divulgación de las investigaciones realizadas.

## Más allá de ser un trabajo para clase, esto se trata de un proyecto real y con perspectiva de continuidad en el futuro.

Cuestión de estilo

## Para el diseño de la página web se ha tomado como referencia el Lynch Lab de Stony Brook, puede consutarse el diseño de la web que se ha tomado como inspiración: <https://www.lynchlab.com/>

## Del Lynch Lab se ha tomado la estructura genérica para la index, así como el diseño minimalista para el texto y la web. Utilizando tonalidades claras, una tipografía especialmente delgada sin serifa y un diseño global que se caracteriza por su neutralidad.

## También se ha tomado como idea varios efectos, como la transparencia del menú al hacer scroll o el attachment:fixed de las imágenes de fondo. Otros efectos, por falta de conocimiento, han resultado imposibles de imitar. Por ejemplo, los videos y animaciones que sirven de fondo han sido remplazados en nuestra web por sombreados como los vistos en clase (box-shadow) pero con la propiedad inset, para dar cierta profundidad (aunque también se han usado box-shadow normales).

## Todo el material fotográfico es propiedad del equipo de investigación del LaPAB.

## Responsive

## Para manejar el responsive de la web se han tomado 3 dimensiones: una primera dimensión que es la genérica, y está pensada para todo tipo de pantallas. Una segunda, para pantallas por debajo de 1050 píxeles, y una tercera para dispositivos móviles (por debajo de 500px).

## Una falta de planificación a la hora de crear el css de la web nos llevó al error de no crear un @media (min-width: 1051px{}), lo cual nos ha obligado a crear etiquetas del tipo .ocultar-big-screen para cambiar el display cuando disminuye el ancho de la pantalla.

## En caso de rehacer la web, o de cara a una futura refactorización, sería necesario especificar qué sucede con ciertos elementos dentro de cara rango, en lugar de compensarlo alterando continuamente la propiedad ‘display’.

## Entre los elementos afectados del responsive, podemos destacar el menú (que para las pantallas más pequeñas es necesario hacer ‘click’ para que se espliegue y pasará de un sticky (por defecto) a una posición estática (y así no moleste en la pantalla). El Aside, que directamente desaparece. El footer, que reorganiza los elementos dentro de un flex. La galería de imágenes, que pasa de un grid a un display:block para evitar que las imágenes queden demasiado pequeñas.

## 

## También cambian los márgenes, que disminuyen a un pequeño porcentaje para evitar que el texto acabe siendo un ‘hilo’ (en pantallas anchas nos interesa que todo el texto quede justificado al centro y haya aire a ambos lados, en pantallas pequeñas preferimos marcar un ‘text-align: left’ para aprovechar mejor el espacio de cada div.

## Grid

## El grid forma parte de una iniciativa completamente experimental para ver cómo funciona. Se ha utilizado en una sola galería de imágenes donde se ha demostrado ser especialmente interesante al controlar exactamente la posición de cada elemento de la galería.

## Para su manejo, se ha establecido un grid de 3x3 (3 columnas y 3 filas) a través de las siguientes propiedades:

grid-template-columns: repeat(3, 1fr);

grid-template-rows: repeat(3, auto);

## Al decir que queremos que ocupe 1fr en la columna, le indicamos al CSS que queremos que cada ‘celda’ (o div anidad) debe ocupar exactamente 1 de las 3 porciones del espacio. En otras palabras, las 3 columnas van a medir lo mismo. Para la altura de cada div anidado (representado en grid-template-rows), el espacio ocupado será automático dependiendo del alto.

## Para dar un paso más allá en el grid, se ha seleccionado un div etiquetado como ‘.large’ y se ha especificado que queremos que se expanda una fila y una columna más allá de la que le corresponde.

grid-column: span 2;

grid-row: span 2;

## 

Puesto que las imágenes están indicadas para ocupar el 100% del div que ocupan (y no hay ni un máximo ni un mínimo), nuestra galería siempre va a ocupar todo el ancho de la pantalla. Esto es un problema según se encoge la pantalla, ya que las fotografías pueden encogerse demasiado. Para solucionar esto, la opción más sencilla ha sido romper el grid y cambiar su display.

Utilización de JavaScript

Para el uso de Javascript hemos decidido jugar con el uso de 3 eventos (en los tres casos son eventos fáciles de manejar y que no involucran una selección múltiple de elementos (lo que nos obligaría a iterar con un forEach() ). En su lugar, hemos apostado por hacer selecciones simples selectElementById(), o seleccionar un bloque entero.

Antes de comenzar, creamos nuestro script dentro de una carpeta que hemos denominado ‘js’, siguiendo la nomenclatura convencional para el backedn que posteriormente podríamos utilizar (en este caso, una hipotética implantación de Django).

Dentro del archivo único de dicha carpeta, especificamos que todo el código que viene a continuación se ejecute después de que se cargue todo el contenido de la página, tal que así:

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

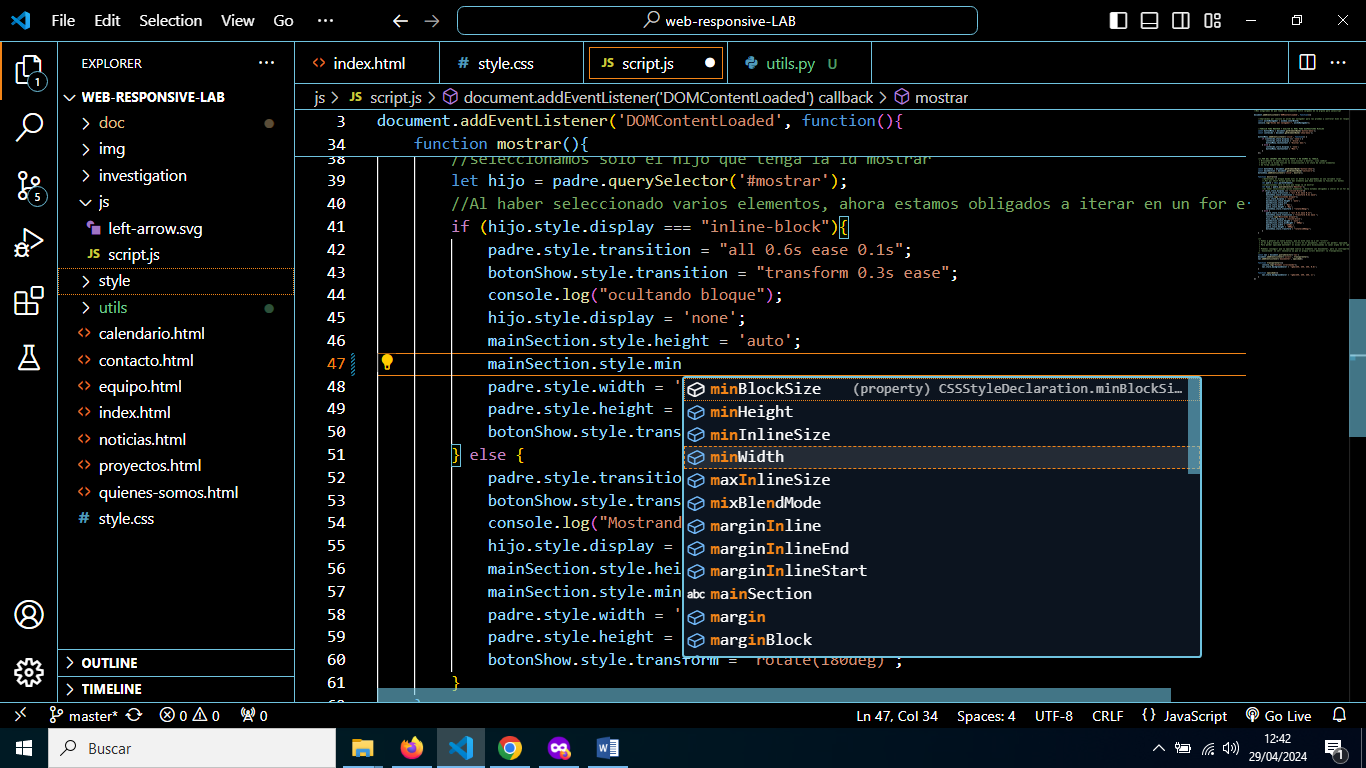
Posteriormente, procedemos a definir nuestras funciones y variables.

Por motivos de legibilidad, y teniendo en cuenta que estamos empezando a usar JavaScript, se ha decidido declarar e inicalizar las constantes únicamente antes de la función que vaya a utilizar dicha constante. El uso de constantes en lugar de variables se debe a que cada elemento seleccionado nunca cambiará su valor (o al menos no está previsto que vaya a cambiar).

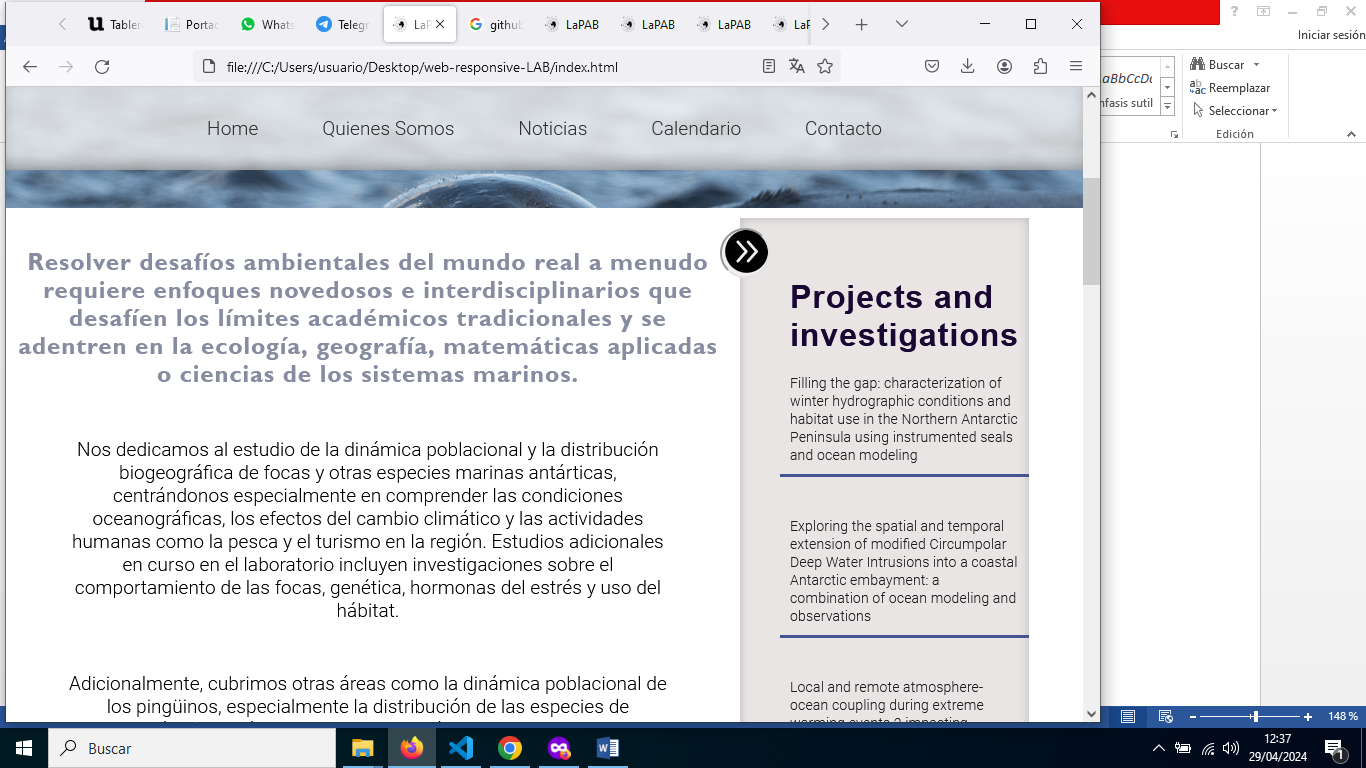
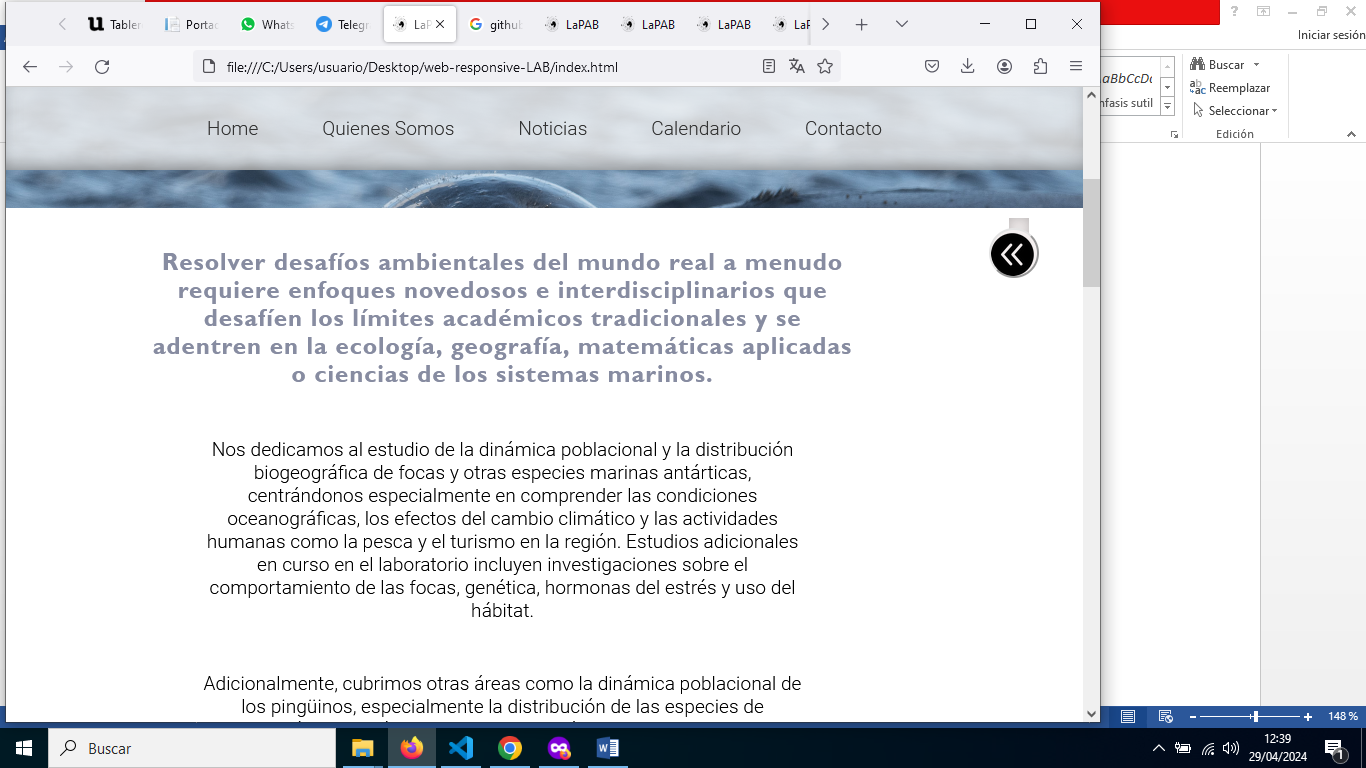
Evento nº1: click

El primer evento, que tiene como selector el primer elemento de la lista del navegador y en otra función un botón del aside, se activará (.addEventListener) al realizar ‘click’, y realizará una sencilla función de cambiar el style de los elementos que hayamos declarado anteriormente como constantes y de las variables locales.

La forma de cambiar la propiedad style no difiere de otros lenguajes de programación, donde a través de un ‘string’ podemos insertar el nuevo valor que queramos asignar, teniendo en cuenta que JavaScript utiliza camel case para nombrar las distintas propiedades que están dentro de Style. Por suerte, el corrector de Visual Studio nos ayuda a renombrar correctamente aquellas que dentro del css utilizan el ‘burrito case’.

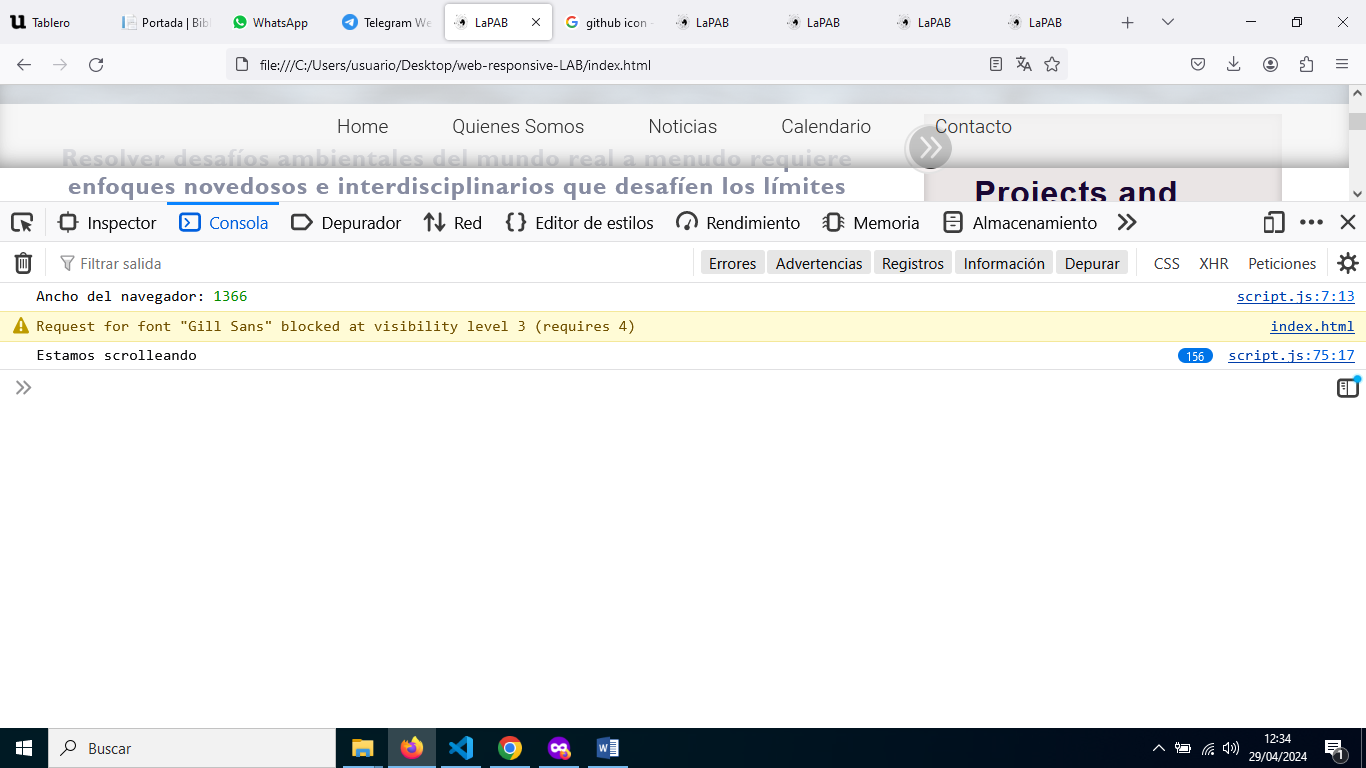


Con el objetivo de reutilizar el código, hemos declarado dentro de una de las funciones un elemento padre e hijo como variables locales (let). De tal forma que en caso de utilizar el botón en otro lugar, pueda seguir funcionando correctamente.

Evento nº2: scroll

El segundo evento, muy similar al anterior, tomará como elemento activador todo el documento. Es decir, en cualquier momento en el que el documento detecte un scroll, se activará la función. La función tomará el valor de la constante nav, y le cambiará el canal alpha para reducir la opacidad.



Evento nº3: mouseenter

Para restaurar la opacidad, hemos implementado la detección de un tercer evento que es el ‘mouseenter’, es decir, cuando el cursor está por encima del elemento activador. Podríamos haber implementado un cuarto evento como ‘mouseleave’, aunque esto no será necesario ya que los eventos relacionados en el scroll son suficientes para manejar esto.

Insertamos mensajes de depuración a lo largo de todo el proceso para depurar el código y poder detectar si hay un fallo detectando el evento o, por el contrario, seleccionado un elemento de la página.

Pueden leerse más detalles sobre Javascript en los propios comentarios de script.js.

Utilización de Python

El laboratorio ha facilitado una gran cantidad de material gráfico, lo que ha supuesto una pequeña complicación a la hora de manejarlo (más aún teniendo en cuenta que en el futuro pueden surgir nuevas fotografías).

Para este caso, se ha creado una carpeta denominada ‘utils’ en la que incluiremos diversas funciones que puedan ser de interés general, como renombrar imágenes de forma masiva o incluso procesar fotografía.

Para este proyecto, únicamente se ha utilizado la librería de ‘os’, que es perfecta para iterar sobre directorios. Aunque posteriormente es posible que necesitemos emplear otras librerías como ‘opencv’ o ‘pillow’. Dada la complejidad que entrañan, dejamos ambas librerías para otra ocasión.