

Final Programación I


Julio 2024

Consignas generales

Para iniciar el examen te daremos una maqueta en html con su correspondiente estilo en CSS y archivos de Javascript. Luego encontrarás una consigna con puntos que deberás desarrollar.

Desde ya **quedan prohibidas** herramientas de comunicación con terceros (Slack, Whatsapp, email, Google Drive, etc). Podrás consultar material de referencia (notas de clase, playground, ejercicios de clase) pero **no podrás consultar** código que tengas hecho previamente en **tu proyecto integrador**.

A lo largo del examen contestaremos **únicamente preguntas sobre la interpretación de las consignas** para poder orientarte pero no así preguntas técnicas o de validación del código.

 Recordá que los puntos de la consigna deben resolverse únicamente con los temas vistos durante la cursada. Los puntos que no respeten esta indicación quedarán anulados.

Al finalizar tu trabajo te pedimos que coloques la carpeta principal dentro de un archivo zip y lo envíes a tu dupla de profesores por slack.

Empecemos...

Pasos preliminares

Dentro del archivo zip encontrarás el proyecto base.

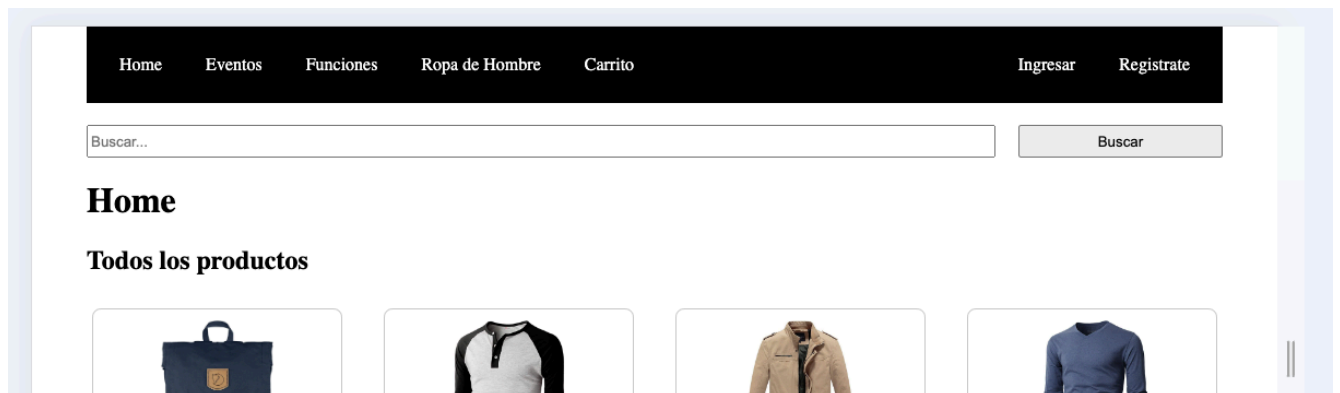
Abrí el archivo zip y copiá todo el contenido del archivo zip en una nueva carpeta y nombrala con el patrón "ApellidoNombre".

Revisá en todos los puntos que los archivos html y js estén correctamente vinculados.

CONSIGNAS

1. CSS y flexbox

En la página principal (index.html) encontrás que el menú de navegación no tiene estilos. Trabajando únicamente en **styles.css** y dentro de los selectores del menú deberás aplicar propiedades de estilo y flexbox para lograr que el menú se vea como en la imagen.



Deberás tener cuenta:

- El color de fondo de toda la barra de navegación es #000. Toda la barra tiene un padding de 10px y un margen inferior de 20px.
- El color de la tipografía es #fff.
- Los elementos "navigation" y "user" comparten las mismas propiedades de flexbox.
- Los li de ambos menús tienen padding de 20px a ambos lados.

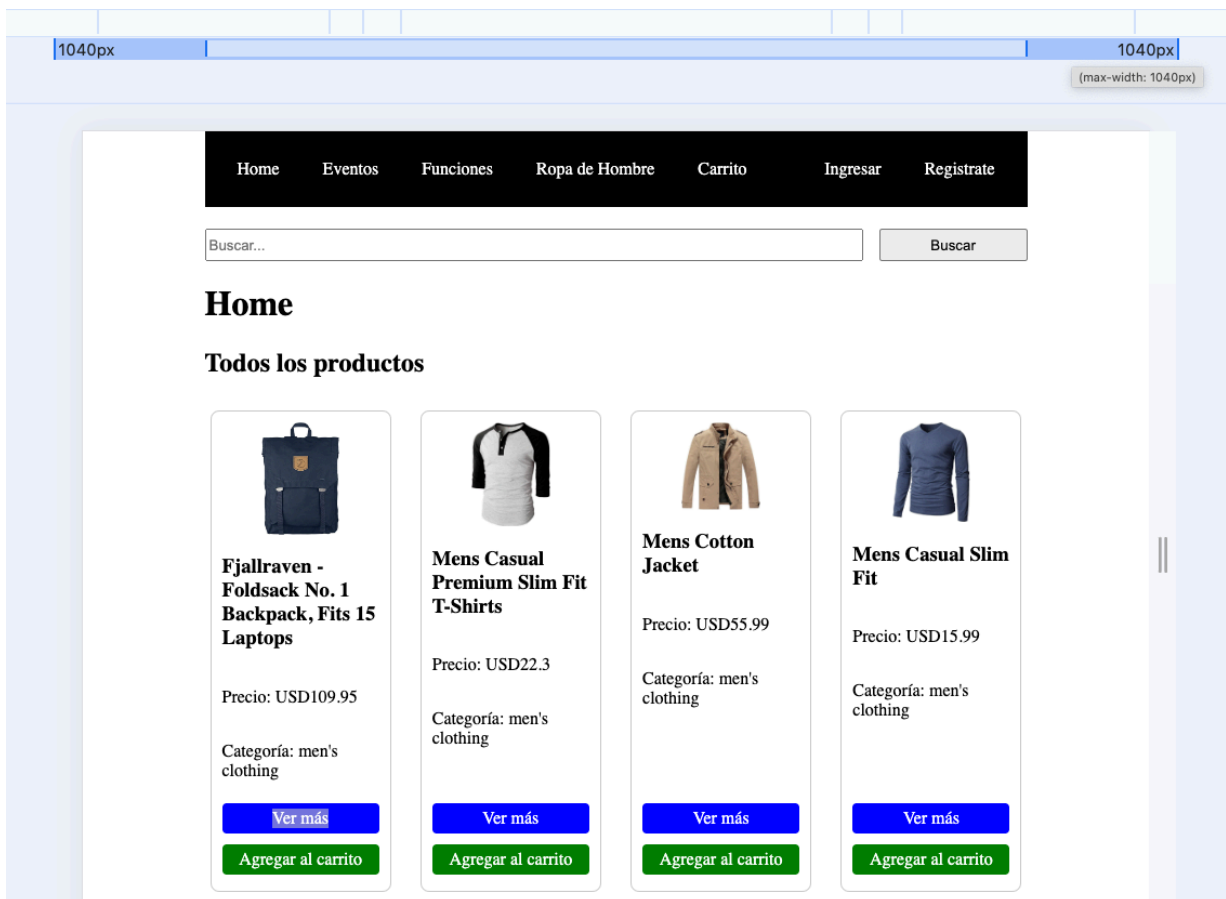
2. Responsive

Para que la página principal (index.html) se vea correctamente en dispositivos pequeños es necesario adaptar el diseño del menú y de las tarjetas de productos.

A) Deberás crear una media query para un display de ancho máximo 1040px que modifique:

- El ancho máximo de la etiqueta body a 760px
- Que los botones "ver más" y "Agregar al carrito" pasen a estar uno arriba del otro.
- Que entre los botones "ver más" y "Agregar al carrito" exista una separación entre si de 5px hacia arriba y hacia abajo.

Si lo resolvés correctamente debería verse así:



B) Deberás crear una media query para un display de ancho máximo 760px que modifique:

- El relleno de la etiqueta body con 10px únicamente hacia ambos laterales.
- Que los menús pasen a estar en columna y con los textos centrados.
- Que a cada elemento del menú les aplique:
 - i) margen superior e inferior de 5px
 - ii) borde de 1px, sólido de color blanco.
 - iii) relleno de 5px en los 4 laterales.
- Que modifique el ancho de las tarjetas de producto a 45%

Si lo hacés correctamente debería quedar así:



Home

Todos los productos



**Fjallraven - Foldsack No. 1 Backpack,
Fits 15 Laptops**

Precio: USD100.05



Mens Casual Premium Slim Fit T-Shirts

Precio: USD22.3

3. Detalle de un producto.

Cada uno de los productos en la página principal del proyecto tiene un link "ver más" que lleva al detalle del producto. Si hacés click en el link "ver más" de alguno de ellos verás que la página de producto está incompleta.

Trabajando únicamente en el archivo **producto.js** deberás vincular el archivo **producto.html** con **producto.js**, obtener el código sku del producto en cuestión y utilizar el endpoint de Fake Store API para conseguir la información. Deberás mostrar la información al usuario dentro de la estructura del archivo **producto.html** utilizando javascript para modificar el DOM. El trabajo de captura y actualización de datos debe ser **individual para cada elemento**.

La información de detalle de cada producto debe coincidir con el producto clickeado en la página principal.

No es necesario agregar etiquetas o clases. Usá **sólo** las que ya se encuentran en el archivo.

La información de los productos es provista por el endpoint

Get a single product

```
fetch('https://fakestoreapi.com/products/1')
```

La documentación está en este link

<https://fakestoreapi.com/docs#p-single>

Si lo resolvés correctamente la página de un producto debería verse así:



4. Eventos

En el archivo **eventos.html** encontrarás el botón verde con el texto "Agregar al carrito". Para comenzar vinculá el html con el archivo **eventos.js**. Trabajando únicamente en el archivo **eventos.js** al clickear sobre el botón deberás reemplazar la estructura html dentro de la section

con clase "single" por una etiqueta h3 con el texto "Muchas gracias por tu compra".

Si lo realizas correctamente la página debería verse así:



5. Formularios y validaciones

En la página principal del sitio (**index.html**) encontrarás un formulario de búsqueda.

Trabajando únicamente **dentro de las etiquetas del formulario** y en **index.js** deberás verificar si el formulario cuenta con todos los elementos html necesarios para funcionar correctamente y llevarte a la página **resultados.html**

Previo al envío el formulario es necesario revisar mediante javascript:

- Que el campo de búsqueda no esté vacío.
- Que si contiene texto tenga al menos 3 caracteres.

🤔 Para ambas validaciones puede ayudarte pensar como medir la longitud de caracteres que escribió el usuario.

Cada una de las validaciones deberá avisarle al usuario lo que debe corregir mostrando un mensaje dentro de la etiqueta <p> con clase "mensaje".

- "El campo no puede estar vacío."
- "El texto debe tener 3 o más caracteres"

Si validás correctamente los mensajes deberían verse así.

El campo no puede estar vacío. ←

Home

Todos los productos

Cumplidas las validaciones el formulario debe llevarte a **resultados.html**

6. Funciones

En el archivo **funciones.js** deberás crear y ejecutar 2 funciones:

- Creá una función llamada **viajarEnElTiempo** que reciba un número como parámetro. La función debe analizar:
 - Si el número ingresado por parámetro es mayor o igual a 55 debe retornar el mensaje "Viajamos a 1955!!".
 - Si el número es menor a 55 retornará "Necesitamos más velocidad".

Ejecutá la función dentro de un `console.log()` para ver el resultado por consola. Si lo hacés correctamente con un número menor a 55 y otro mayor a 55 deberías ver los textos en la consola.

```
Necesitamos más velocidad
```

```
Viajamos a 1955!!
```

- Dentro del archivo **funciones.js** encontrarás un array de objetos literales con algunos personajes de Game of Thrones. Tu trabajo consiste en crear la función **soloNombres()** que reciba un array como parámetro. La función debe recorrer el array ingresado como parámetro y retornar un nuevo array que contenga únicamente los nombres de los personajes. Ejecutá la función dentro de un **console.log()** para ver el resultado por

consola. Si lo hacés correctamente deberías ver el siguiente array adentro del siguiente array

```
▶ (4) ['Jon', 'Daenerys', 'Arya', 'Tyrion']
```