Vistas y elementos dinámicos

En estas actividades vas a renderizar la página de detalles modificando el DOM con JavaScript. Analiza el mockup de la vista provisto ya que debe quedar lo más parecido posible.

> 💡 ***Atención***
>
> *Antes de finalizar su sesión en equipo de hoy, un integrante deberá compartir su pantalla y explicar en detalle cómo resolvió los ejercicios propuestos. Puede ser el facilitador o cualquier compañero que se anime (puedes ser tú 💪).*
>
> *Aprovecha esta oportunidad para enriquecer tu propia resolución y participa activamente haciendo preguntas.*

🖍️ Actividad: Renderizar dinámicamente la vista de detalle

Para este ejercicio tienes que seguir los siguientes pasos:

1. Desde Visual Studio Code, abre la carpeta **store** y luego el archivo productDetail.js
2. Define la función printDetails para que:
    1. dependa del parámetro id del detalle del producto
    2. guarde en una variable el producto encontrado con el id del parámetro
    3. defina una variable para la plantilla de la vista de detalles e interpole con ${} los datos correspondientes al detalle del producto
    4. seleccione un id del html donde se imprimirá la plantilla
    5. actualice la vista con la renderización del detalle

A modo de ejemplo:

```
function printDetails(id) {
   const product = products.find((each) => each.id === id);
   const detailsTemplate = `
      //pegar todo el contenido de la vista de detalle
      //identificar los datos estáticos
      //interpolarlos con ${}
   `;
   const detailsSelector = document.querySelector("#details");
   detailsSelector.innerHTML = detailsTemplate;
}
printDetails(id)
```

1. Abre con live server el archivo index.html el archivo index.html, prueba el direccionamiento de los links y verifica la correcta renderización de la vista de detalle.

2. Programa y verifica la vista la cantidad de veces que sea necesario para que la vista se renderice correctamente.

🖍️ Actividad: Renderizar dinámicamente el select de color

En esta actividad, sólo nos enfocaremos en renderizar dinámicamente las opciones del select de colores del producto. Para este ejercicio tienes que:

1. Desde Visual Studio Code, abre la carpeta **store** y luego el archivo productDetail.js

2. Modifica la función printDetails para que el select renderice de forma dinámica las opciones de los colores.

3. Luego de “mapear”, une todos los elementos del array transformado con join() para retornar un único template. A modo de ejemplo:

```
function printDetails(id) {
   const product = products.find((each) => each.id === id);
   const detailsTemplate = `
      ...
      <select type="text" placeholder="Selecciona un color">
         ${product.colors.map(
            (each) => `<option value=${each}>${each}</option>`
         ).join("")}
      </select>
      ...
   `;
```

```
   const detailsSelector = document.querySelector("#details");
   detailsSelector.innerHTML = detailsTemplate;
}
printDetails(id)
```

1. Programa y verifica la vista la cantidad de veces que sea necesario para que la vista se renderice y funcione correctamente.

🖍️ Actividad: Renderizar dinámicamente las miniaturas

En esta actividad, sólo nos enfocaremos en renderizar dinámicamente las miniaturas del producto. Para este ejercicio tienes que:

1. Desde Visual Studio Code, abre la carpeta **store** y luego el archivo productDetail.js

2. Modifica la función printDetails para que la caja de las miniaturas renderice correctamente la cantidad de fotos que tenga el array de fotos del producto.

3. Luego de "mapear", une todos los elementos del array transformado con join() para retornar un único template. A modo de ejemplo:

```
function printDetails(id) {
   const product = products.find((each) => each.id === id);
   const detailsTemplate = `
      ...
         <div class="product-images-block">
            ${product.images.map(
                each => `<img class="miniImg" src="${each}" alt="mini" />`
            ).join("")}
         </div>
      ...
   `;
   const detailsSelector = document.querySelector("#details");
   detailsSelector.innerHTML = detailsTemplate;
}
printDetails(id)
```

1. Programa y verifica la vista la cantidad de veces que sea necesario para que la vista se renderice y funcione correctamente.

> 💡 *Procurar que las "columnas" con las fotos, los detalles del producto y el cuadro de añadir tengan el mismo ancho (ideal 360px).*

📖 Repositorio en GitHub

Puedes subir a un repositorio existente de github la práctica del día de hoy. Los comandos básicos son:

- **git add .** (para preparar todos los archivos creados, modificados o eliminados)
- **git commit -m "nombre del envío"** (para versionar en repositorio local)
- **git push origin main** (o master según corresponda para enviar la nueva versión del proyecto al repositorio remoto)