Este ejercicio de CSS Grid debes subirlo a tu Git-Hub para poder revisarlo

Y por último, las preguntas teóricas son:

1. **¿Cuáles son algunas de las cosas que hacen que SCSS sea diferente de CSS? (coloca ejemplos)**
   El SCSS tiene extensión .sass, y el CSS, extensión .css. El SCSS se parece en la sintaxis al CSS, lo cual es ventajoso, ya que el CSS es el código que los navegadores entienden; sin embargo hay algunas diferencias: el SCSS tiene indentaciones, anidamientos y algunos símbolos diferentes, como el selector & que sirve para llamar al mismo selector que estamos usando, ahorrando escribir el selector varias veces. **Ejemplo**:

CSS:

```
.Item1 p {
  margin: 10;
  font-size: 1.1rem;
}

.Item1 ul {
   Color: blue;
}

.Item1 a {
   text-decoration: none;
 }
```

SCSS:

```
.Item1 {
  p {
    margin: 10;
    font-size: 1.1rem;
  }

  ul {
    Color: blue;
  }

  a {
     text-decoration: none;
   }
}
```

La diferencia más importante es que el SCSS puede ser más dinámico, más simple y optimiza el CSS, al poder utilizar variables, anidamientos o nestings (como muestra el recuadro anterior), grupos de propiedades que se llaman mezclas o “mixin” e incluso *mixins* con argumentos, lo que le hace aún mucho más flexible que el CSS, y además puede importar ficheros. Esto a su vez, hace que el diseño sea ‘más divertido’.

2. **¿Qué es una variable SCSS? (porque crees que debes utilizarla pon un ejemplo de una variable, escribe una variable y como se pondría para utilizarla)**

   Las variables almacenan información que necesitamos reutilizar en nuestro código. En lugar de repetir valores, asignamos un valor al nombre de la variable (poniendo por delante el símbolo de **$**) y luego podemos referirnos a esa variable en lugar de referirnos al valor. Esto evita

constantes repeticiones; hace que cuando queramos modificar las líneas de código, no necesitamos buscar cada una de las líneas afectadas y cambiar una a una: lo que haríamos es modificar las variables y ya está. Hace que el código sea más fácil y rápido de manejar.

3. **¿Qué es un SCSS Mixin? (porque crees que debes utilizarla pon un ejemplo de un mixin, escribiendo cómo se crea y como se pondría para utilizarla)**

Un *mixin* (@mixin) es un conjunto de propiedades que se incluyen en un selector, y que se van a reutilizar en el código. Lo reclamamos con @include. Un ejemplo: algunas de las propiedades que utilizaré frecuentemente en un encabezado, añadiendo otras que no deseo incluir en el *mixin* porque no es probable que las repita:

Ejemplo de mixin:

```
@mixin flex-center ($header-color: Gray) {
  display: flex;
  justify-content: center;
  align-items: center;
  color: $header-color;
}


.header {
  @include flex-center;
  height: 60px;
  background-color: #f6f6f6;
}
```

He añadido un argumento (se puede añadir más de uno) porque quizá la mayor parte de las veces me interesa que mi encabezado tenga el color ***Gray***, pero por ejemplo alguna vez puedo necesitar que sea del color ***Blue***.
Dado que en este caso no he puesto argumento en el ***@include***, el selector ***.header*** aplicará el color ***Gray*** por defecto; cuando necesite que sea azul, añadiré el argumento: **@include flex-center (Blue);,** que modificará a ***Blue***.

4. ¿Qué significa Unidad fraccionaria (fr) con CSS Grid?

Fr viene de 'fraction', y representa una fracción o una parte del **espacio disponible** del *grid container*. Es muy flexible y útil porque podemos establecer el ancho de una columna (o una fila) con relación a las columnas (o filas) contiguas.

Por ejemplo, si tengo 3 columnas, puedo hacer que una de ellas (la última) tenga anchura doble que las otras **(grid-template-columns: 1fr 1fr 2fr;)**. Esto significaría que en total tendremos **4fr**, por lo que para averiguar cuántos píxeles son asignados a **1fr**, deberemos dividir el total de píxeles entre 4. Se puede mezclar con **px:**

Por ejemplo:

```
.grid {
        display: grid;
        width: 1000px;
        grid-template-columns: 1fr 310px 2fr 2fr;
        gap: 20px;
}
```

Son 4 columnas, por tanto en total son 3 **gaps = 60px**.
Lo ya asignado entonces sería 310px (2ªcolumna) y los 3 gaps que hay al ser 4 columnas =60px.
Un total de **370px.**

Para saber cuántos **px** es una unidad fraccionaria (**1fr**) averiguaremos cuánto espacio nos queda disponible después de lo ya asignado:
Espacio total (**1000px**) menos lo asignado (**370px**) = **630px disponibles**.

Para saber el valor de **1fr**, dividiremos ese espacio disponible 630px entre el total de 5 unidades asignadas (1fr+2fr+2fr).

1fr = 630/5 = **126px**