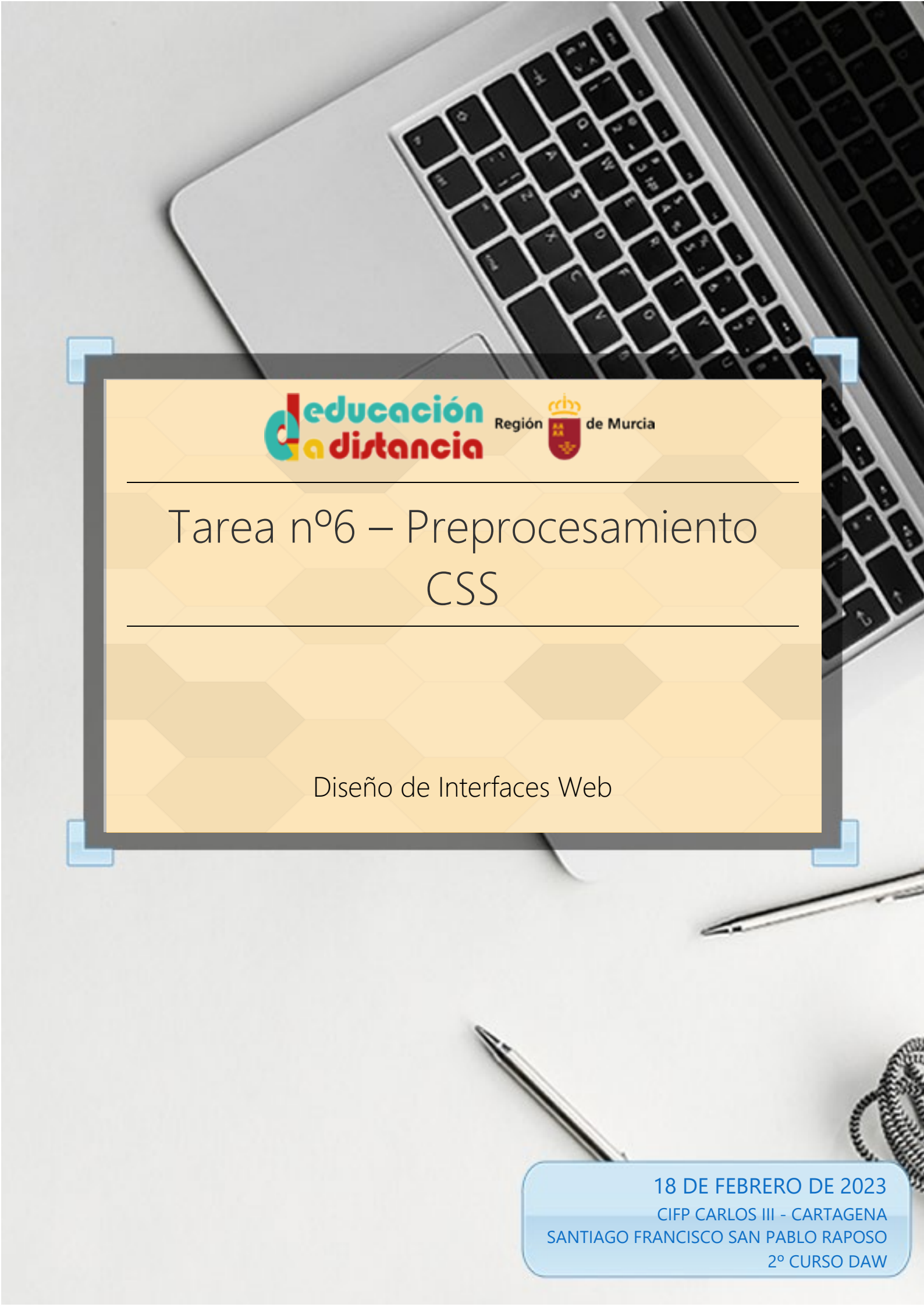


Tarea nº6 – Preprocesamiento CSS

Diseño de Interfaces Web

18 DE FEBRERO DE 2023
CIFP CARLOS III - CARTAGENA
SANTIAGO FRANCISCO SAN PABLO RAPOSO
2º CURSO DAW

Contenido

Tarea para DIW06.

1.- Apartado 1 (5 puntos).

Utilizando como plantilla el código html superior:

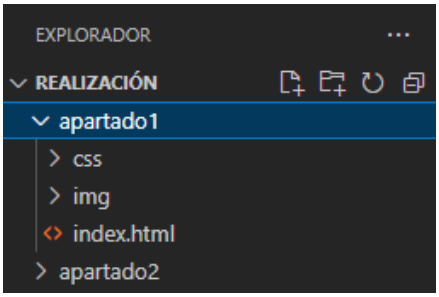
- Crea una nueva carpeta en Visual Studio Code llamada apartado01 y añade el archivo index.html con el código proporcionado.
- Abre en el navegador la ruta del archivo index.html y comprueba que funciona correctamente.
- Crea las carpetas css e img.
- Crea, dentro del carpeta css, el fichero style.scss.
- No debes escribir nada en el archivo style.css, todo el código de dicho archivo se compilará partir del fichero style.scss. Además, no puedes utilizar directivas de tipo @, salvo @import para la fuente de Google.
- Accede a Google Fonts e importa la fuente Nunito en sus variantes light 300, regular 400 y bold 700 dentro del archivo style.scss.
- Enlaza el archivo style.css en tu archivo index.html.
- Modifica tu hoja de estilos para que la página y las fuentes cumplan los siguientes requisitos (intenta utilizar todo lo aprendido hasta ahora para optimizar tu hoja de estilos):
    - Ajusta la página a un ancho máximo de 1400px sin márgenes y céntrala en la ventana del navegador.
    - La fuente en toda la página será Nunito y la alternativa será sans-serif.
    - La fuente del cuerpo de la página tendrá un peso de 400 con un tamaño de 1rem.
    - La fuente de los titulares (h1 a h3) llevará un peso de 700 y su tamaño será mayor que el cuerpo en cada paso (un 50% más para h3, un 100% mas para h2, un 200% mas para h1)
- En este punto, al refrescar tu página deberías ver los cambios en las fuentes. Si no es así, revisa la configuración y los estilos antes de continuar.
- Modifica tu hoja de estilos para que el **header** cumpla los siguientes requisitos.
    - El logo de Sass (que puedes descargar de su página web en svg), debe ir alineado a la izquierda ocupando el 10% del ancho del header.
    - El título de la página debe aparecer centrado en el espacio restante del header, y el subtítulo debajo y también centrado.
    - Pinchando en la imagen o en el título la página se debe recargar, pero el título no debe parecer un enlace (de hecho, ningún enlace de la página debe parecerlo).
    - Resetea los márgenes de los títulos del header (ponerlos a 0).
    - Añade al header un color de fondo y un pequeño padding a los cuatro lados.
- Modifica tu hoja de estilos para que el **nav** cumpla los siguientes requisitos:
    - El nav ocupará todo el ancho disponible, tendrá el mismo color de fondo y estará levemente separado del header.

    - Los ítems de menú deben estar en línea y centrados en pantalla, sin margin y con un mínimo padding entre ellos.
    - Los ítems de menú estarán en mayúsculas, con un tamaño de letra del 110% respecto del cuerpo, un peso de 400 y uno de 700 on hover.
    - El color de los ítems y los enlaces será negro, salvo en los enlaces on hover que será rojo.
- Aplica a la section una altura mínima de 20em para separarla y que se diferencie del footer. Ya trabajaremos con ella más adelante.
- Modifica tu hoja de estilos para que el footer cumpla los siguientes requisitos:
    - Las dos secciones del footer (div y p) tendrán el mismo color de fondo y estarán separadas levemente.
    - Descárgate los iconos de las redes sociales en svg de ESTA DIRECCIÓN.
    - Asócialos con sus correspondientes enlaces en el html.
    - Los iconos del footer tendrán un ancho de 3em, estarán centrados y en línea, con una cierta separación entre ellos.
    - La letra del párrafo p será un 80% respecto de la del cuerpo.

Por lo que llevas en la práctica tendrás que tener un aspecto parecido a este:

Solución propuesta para el apartado nº 1

Para la realización de este apartado, se ha creado un proyecto en Visual Studio Code dentro del cual, hay dos carpetas. Una de ellas es la del apartado 1, en la cual se almacena el index.html. Dentro de una subcarpeta "css" el fichero .scss (basado en Sass) y el fichero CSS con el que al final el HTML se enlaza. También, dentro de una subcarpeta llamada img, están las imágenes que se utilizan dentro del HTML.



De esta forma, **mantenemos intacto el HTML** que nos da el enunciado por defecto, y solamente trabajo con la hoja de estilo SCSS, que me genera, gracias a la extensión Live Sass Compiler (disponible para descargar dentro de la Marketplace de Visual Studio Code), el

archivo compilado ya en CSS al que hace referencia el archivo HTML. La única modificación que hacemos en el HTML es para enlazar con la ruta correcta al CSS.

Tal y como se puede observar en el código de SCSS (adjunto junto a la memoria en el archivo comprimido), en este apartado se hace uso única y exclusivamente de la directiva @import para poder importar la fuente "Nunito" desde Google Fonts.

```scss
/*-- Import --*/
@import url('https://fonts.googleapis.com/css2?family=Nunito:wght@300;400;700&display=swap');
```

Respecto a los requisitos de las fuentes: yo lo he resuelto de la siguiente forma: me he definido una serie de variables, que luego son las que uso dentro del body para aplicar esos estilos de forma general a toda la página:

```scss
/*-- Variables --*/
$t-letra: 1rem;
$peso-cuerpo: 400;
$peso-negrita: 700;
$color-fondo: rgba(128, 128, 128, 0.368);
$color-titulo: black;
```

```scss
/*-- Estilos --*/
body {
    max-width: 1400px;
    margin: 0 auto;
    font-family: 'Nunito', sans-serif;
    font-size: $t-letra;
    font-weight: $peso-cuerpo;
```

Se puede apreciar en esta última imagen como he aprovechado para definir directamente sobre el body el **tamaño máximo de la página** y el margen automático por los laterales. Este ancho máximo luego será trasladado a una variable en el apartado 2.

Para las **fuentes de los titulares (h1 a h3)**, he configurado las propiedades apropiadamente para cada uno de los tres tipos de encabezados (h) que necesitamos:

```scss
    h1 {
        // font-weight: calc($peso-titulos * 3);
        font-weight: $peso-negrita;
        font-size: calc($t-letra * 3);
    }

    h2 {
        // font-weight: calc($peso-titulos * 2);
        font-weight: $peso-negrita;
        font-size: calc($t-letra * 2);
    }

    h3 {
        // font-weight: calc($peso-titulos * 1.5);
        font-weight: $peso-negrita;
        font-size: calc($t-letra * 1.5);
    }
}
```

En el header, de acuerdo con lo que pide el enunciado, he hecho que se cumplan las siguientes propiedades, utilizando para ello las variables que definí antes cuando procedía:

```
38  header {
39      a {
40          img {
41              width: 10%;
42              float: left;
43          }
44
45          h1 {
46              text-align: center;
47              margin: 0;
48              color: $color-titulo;
49          }
50      }
52      h2 {
53          text-align: center;
54          margin: 0;
55      }
56
57      background-color: $color-fondo;
58      padding: 1%;
59  }
```

Podemos ver que, hasta ahora, cumple todo lo que pide el enunciado: logo Sass ocupando el 10% del ancho del header, título y subtítulo centrados, enlace "camuflado" (ya que le eliminamos la apariencia natural de los enlaces), márgenes establecidos a 0, color de fondo del header, y padding.

Sin embargo, como el enunciado dice que ningún enlace de la página debe "parecerlo", he decidido sacar la esta propiedad de forma general para cualquier etiqueta "a" dentro del documento HTML:

```
61  a {
62      text-decoration: none;
63  }
```

En el nav: no he tenido que tomar ninguna decisión de diseño más allá que ir siguiendo lo que me pedía el enunciado, utilizando por supuesto, las variables que definimos inicialmente. Reseñar únicamente que para lograr centrar horizontalmente los enlaces del nav, hemos utilizado la propiedad text-align: center sobre el padre que los contiene (en este caso, sobre ul). Este es el código que da formato al nav:

```scss
nav {
    background-color: $color-fondo;
    margin-top: 1%;
    width: 100%;

    ul {
        text-align: center;
        li {
            display: inline-block;
            margin: 0;
            padding: 1%;
            text-transform: uppercase;
            font-size: calc($t-letra * 1.1);
            font-weight: $peso-negrita;

            a {
                color: black;
                font-weight: $peso-cuerpo;

                &:hover {
                    font-weight: $peso-negrita;
                    color: red;
                }
            }
        }
    }
}
```

A la sección (section), le hemos aplicado sin más la altura mínima que nos pedía:

```scss
section {
    min-height: 20em;
}
```

Respecto al footer: igual que en el nav, conseguimos centrar los elementos con la propiedad text-align: center aplicada sobre el padre (es decir, sobre footer, que es quien los contiene).

Además, agrupamos en un mismo selector las propiedades que son comunes a div y p, dada la estructura del fichero HTML, y para el div en específico, le asignamos a las imágenes que contiene el ancho de 3em que nos pide el enunciado. El tamaño de la letra del párrafo se calcula sobre la variable $-letra (que es la que se aplica al resto del cuerpo) multiplicando por 0,8.

```
97  footer {
98      text-align: center;
99      div, p {
100         background-color: $color-fondo;
101         margin-top: 1%;
102     }
103
104     div {
105         img {
106             width: 3em;
107         }
108     }
109
110     p {
111         font-size: calc($t-letra * 0.8);
112     }
113 }
```

2.- Apartado 2 (5 puntos).

Partiendo del apartado01, crea una copia en una nueva carpeta llamada apartado02.

- Abre en el navegador la ruta del archivo index.html y comprueba que funciona correctamente.
- **Traslada todas las variables de la hoja de estilos a una nueva hoja llamada _variables.scss** e impórtala en la actual.
- **Modifica tu hoja de estilos para que cumpla los siguientes requisitos**:
    - El color del título de la página debe venir de una @function escrita en _variables.scss que genere un color aleatorio al compilar de componente rojo (0 en el resto).
    - El color del subtítulo de la página debe venir de una @function escrita en _variables.scss que genere un color aleatorio al compilar de componente verde (0 en el resto).
    - Simplifica las reglas de estilo de fuentes de los títulos mediante un @mixin, @include y @extend.
    - Los estilos de lista horizontal del nav (salvo los enlaces) deben ser reutilizables mediante un @mixin.
- Modifica tu hoja de estilos para que la **section** cumpla los siguientes requisitos:
    - Añade 6 article en columnas de 3.
    - Añade a cada article un enlace a la red social (de las del footer), su logo a la izquierda, un título a la derecha y una descripción de la empresa debajo.
    - Cada article tendrá un color de fondo aleatorio, con opacidad de 0.5 y diferente de los otros (puedes utilizar como punto de partida la @function definida anteriormente).
    - El logo del article tendrá un ancho del 25% y el resto del contenido ocupará el espacio restante del article.
- Modifica tu **hoja de estilos para adaptarla a diferentes dispositivos**:
    - En pantallas de menos de 992px, los articles pasan a columnas de 2, en pantallas de menos de 768px, aparecen uno debajo de otro.
    - En pantallas de menos de 768px, el logo aparece centrado encima del título y el subtítulo.
    - En pantallas de menos de 768px, el margen de separación entre secciones desaparece.
    - En pantallas de menos de 768px, los ítems del nav aparecen uno debajo de otro (quedando así preparados para implementar un menú oculto).
    - En todos los casos, las fuentes e imágenes se adaptan según las necesidades.

2.1.- Solución propuesta al apartado 2.

Para resolver este apartado, me ha llevado crear el SCSS bastante tiempo, y tras bastantes procesos de refinamiento del código, he llegado a una estructura que realmente me ha convencido, y creo que puede ser útil de cara a poder reutilizarla en otros proyectos de SCSS de forma que me permita acelerar bastante mi trabajo.

En el principio del style.css correspondiente al apartado 2, tenemos, igual que en el apartado 1, la importación de la fuente “Nunito” desde la web de Google, y a continuación, llama la

atención que, pese a que el enunciado nos insta a crear un Partial para las variables, yo he creado dos: uno para la maquetación, y otro para las variables en sí propias de este diseño.

Además en ese orden, ya que el Partial _variables podrá usar las variables básicas que definamos en el Partial _maquetacion para el tamaño de nuestra página web, el nº de columnas que queremos que tenga nuestro sistema de maquetación para el caso particular, o el margen (vertical) que se quiera dejar entre bloques.

```
/*-- Import --*/
@import url('https://fonts.googleapis.com/css2?family=Nunito:wght@300;400;700&display=swap');
@import '_maquetacion';
@import '_variables';
```

Me explico:

Como en el final del apartado, el enunciado nos especifica que la página debe adaptarse a distintos tamaños de pantalla, esto nos recuerda al diseño responsive que usaba Bootstrap, por lo que he decidido crearme mi propia "plantilla de Bootstrap" en SCSS, mediante el Partial que se llama "_maquetacion".

Este Partial contiene:

- **Variables básicas de tamaño de página, columnas y márgenes**:

```
/*-- Variables para configuración de disposición de los elementos (maquetación) --*/
$base-ancho: 1400; // Modificable
$ancho-maximo: $base-ancho * 1px;
$divisionColumnas: 12; // Modificable
$columna: calc($base-ancho / $divisionColumnas);
$pcolumna: calc($columna / $base-ancho * 100%);
$margen-between-bloques: 1%;
```

- **Variables para los puntos de ruptura** (que usaremos en el Partial de _variables, de ahí la importancia del orden en el que importamos los Partials).
- **Una única regla CSS**: para incluir en el tamaño de caja el borde que esta pueda tener.

```
* {
    box-sizing: border-box;
}
```

- **Mixin "wrapper"** para poder definir una etiqueta en el SCSS como contenedor principal (que en el caso de esta tarea, es el propio body, ya que no podemos alterar el fichero HTML para añadir un div wrapper).

```
/*-- Mixins para maquetar --*/
@mixin wrapper {
    max-width: $ancho-maximo;
    width: 100%;
    margin: 0 auto;
}
```

- **Mixin "row"**: para poder definir filas, al igual que en Bootstrap. Se ha optado por utilizar el sistema Flex para simplificarlo todo y hacer que se parezca más a Bootstrap en cuanto a comportamiento se refiere.
    - **Además, contiene una serie de parámetros opcionales** relacionados con el posible borde que podemos establecerle a la etiqueta en la que se importe este mixin: sizeBorder, colorBorder y border-style.

```
28  @mixin row($sizeBorder: 0, $colorBorder: white, $border-style: none) {
29      // clear: both;
30
31      display: flex;
32      flex-direction: row;
33      flex-wrap: wrap;
34
35      /*-- Borde --*/
36      border-color: $colorBorder;
37      border-style: $border-style;
38      border-width: $sizeBorder;
39  }
```

- **Mixin "col"**: que permite establecerle a un elemento contenido en un "row" el nº de columnas que se expandirá (es decir, el nº de columnas que ocupará en el espacio de una fila, de las 12 que establecimos inicialmente en las variables del principio del Partial de maquetación).
    - **Además, contiene una serie de parámetros opcionales** relacionados con el posible borde que podemos establecerle a la etiqueta en la que se importe este mixin: sizeBorder, colorBorder y border-style. Si queremos especificar estos parámetros de configuración del borde interior, debemos pasarle al mixin un dato más: el nº de elementos "col" que hay dentro del "row".

```scss
@mixin col($span, $sizeBorder: 0, $colorBorder: white, $border-style: none, $nElementosQueAfecta: 1) {
    /*-- Tamaño ancho de la columna, definido por $i --*/
    /*-- Código viejo --*/
    // float: left;
    // display: inline-block;

    width: calc($pcolumna * $span);

    /*-- Configuración de los bordes --*/
    @if ($sizeBorder != 0) {
        /*-- Definimos propiedades generales para todos los elementos que apliquen el mixin --*/
        border-color: $colorBorder;
        border-style: $border-style;
        border-top-width: 0;
        border-left-width: 0;

        /*-- Personalizamos bordes internos --*/
        @for $i from 1 through $nElementosQueAfecta { //nColumnas
            &:nth-child(#{$i}) {
                $columnas: calc($divisionColumnas / $span); //Define el nº de columnas que se mostraran para el tamaño normal
                @if ($i % $columnas > 0) {
                    border-right-width: $sizeBorder;
                } @else {
                    border-right-width: 0;
                }

                @if (floor(calc(($i - 1) / $columnas)) != (calc($nElementosQueAfecta / $columnas - 1))) {
                    border-bottom-width: $sizeBorder;
                } @else {
                    border-bottom-width: 0;
                }
            }
        }
    } @else {
        border-style: none;
        border-width: 0;
    }
}
```

En el caso de nuestra tarea, este número se almacena en el Partial _variables (nuevamente, se ve la relación que hay entre ambos Partial), y es dentro del fichero .SCSS principal donde se utiliza esta variable para incluir (con la directiva @include) este mixin de col:

```scss
section {
    @include row;
    @include margin-top-bloques;
    min-height: 20em;

    article {
        @include col(4, 4px, white, solid, $numArticulos);
        // @include col(4);
        @for $i from 1 through $numArticulos { //nColumnas
            &:nth-of-type(#{$i}) {
                background-color: colorRandom();
            }
        }
```

En esta captura también se puede apreciar **cómo se ha resuelto el problema del fondo de color aleatorio diferente para cada artículo**: utilizamos & para referirnos al selector padre (article), y con nth-of-type, buscamos el elemento de la iteración en que se encuentre el bucle. Esto se indica mediante la interpolación #{$i}, que variará en cada iteración del bucle for. **La interpolación nos permite "inyectar" texto proveniente de SASS a los selectores CSS**.

También se ha utilizado el mixin “col” nuevamente para el resto de tamaños de pantalla, junto al resto de “modificaciones” visuales en otras partes del HTML que pedía el enunciado para cada uno de ellos:

```scss
@media screen and (max-width: $lg) { //max 992px
    section {
        article {
            @include col(6, 4px, $color-fondo-pagina, solid, $numArticulos);
        }
    }
}
```

```scss
@media screen and (max-width: $md) { //max 768px
    header {
        text-align: center;
        img {
            float: none;
        }
    }

    section {
        article {
            @include col(12);
        }
    }

    nav {
        @include navVertical;
    }

}
```

Ya hemos visto que se utilizan algunos mixin y funciones de los que no hemos hablado todavía: eso es porque están localizados en el Partial _variables, el cual contiene la implementación de estas características específicas de este sitio web:

```scss
/*================================
        VARIABLES
==================================*/
/*-- Variables para las fuentes y colores de letra --*/
$letra: 'Nunito', sans-serif;
$t-letra: 1rem;
$peso-cuerpo: 400;
$peso-negrita: 700;
$color-fondo-pagina: white; // Explicar en la memoria
$color-fondo: rgba(128, 128, 128, 0.368);
// $color-enlaces-nav: black;
$color-enlaces: black;
$color-enlaces-nav-hover: red;

/*-- Variables para configurar el contenido --*/
$numArticulos: 6;
```

Por ejemplo, para el color del título y el subtítulo, se han creado previamente en este Partial dos mixin:

```scss
/*================================================
        FUNCIONES Y MIXINS ESPECIFICOS
================================================*/

/*-- Funcion para el color de rojo aleatorio de título de la pagina --*/
@function colorRojoRandom() {
    @return rgb(floor(random(256)), 0, 0);
}

/*-- Funcion para el color de verde aleatorio de subtitulo de la pagina --*/
@function colorVerdeRandom() {
    @return rgb(0, floor(random(256)), 0);
}
```

Por otra parte, la **simplificación de las reglas de estilo de fuentes de los títulos** que nos pide el enunciado, también la hemos hecho con un mixin:

```scss
/*-- Mixin para reglas de estilo de fuentes de los títulos --*/
@mixin titulos($nivel) {
    font-weight: $peso-negrita;

    @if ($nivel == "h1") {
        font-size: calc($t-letra * 3);
    } @else if ($nivel == "h2") {
        font-size: calc($t-letra * 2);
    } @else if ($nivel == "h3") {
        font-size: calc($t-letra * 1.5);
    }
}
```

Así como también, los **estilos de la lista horizontal del nav** han sido unificados en un mixin, del que hereda **también el mixin "navVertical"**, usado para la visualización del sitio web en pantallas pequeñas:

```
62  @mixin navHorizontal {
63      background-color: $color-fondo;
64      @include margin-top-bloques;
65      width: 100%;
66
67      text-align: center;
68      ul {
69          margin: 0;
70
71          li {
72              display: inline-block;
73              margin: 0;
74              padding: 1%;
75              text-transform: uppercase;
76              font-size: calc($t-letra * 1.1);
77              font-weight: $peso-negrita;
78          }
79      }
80  }
81
82  @mixin navVertical {
83      @include navHorizontal;
84      ul {
85          li {
86              display: block;
87          }
88      }
89  }
```

La **función colorRandom** que se ha visto antes utilizar en la captura en la que se explicaba el uso del mixin "col" sobre las etiqueta "article" dentro de "section", también se encuentra definida dentro de este Partial _variables:

```
/*-- Funcion para el color aleatorio de los articles --*/
@function colorRandom() {
    @return rgba(floor(random(256)), floor(random(256)), floor(random(256)), 0.5);
}
```

Para lograr que el logo del article tenga un ancho del 25% y el resto del contenido, ocupe el espacio restante, se hace de esta forma: ¼ de 12 (columnas que tiene nuestro sistema de maquetación) son 3 columnas, por tanto, asignamos con el mixin col 3 columnas a la imagen, y las 9 restantes al div que contiene la información de la red social de la que trata el article:

```
91          a {
92              @include row;
93              img {
94                  @include col(3); //25%
95                  // width: 25%;
96                  // display: inline-block;
97              }
98
99              div {
100                 @include col(9); //75%
101                 // width: 75%;
102                 // display: inline-block;
103             }
104         }
```

Por último, para la separación entre secciones, se utilizan los siguientes mixin, también definidos en el Partial _variables, que además, ya tienen implementada la funcionalidad para que en pantallas de menos de 768px, el margen de separación entre secciones desaparezca.

```
45  /*-- Mixins para agregar espacio entre bloques --*/
46  @mixin margin-top-bloques {
47      margin-top: $margen-between-bloques;
48
49      @media screen and (max-width: $md) {
50          margin-top: 0;
51      }
52  }
53
54  @mixin margin-bottom-bloques {
55      margin-bottom: $margen-between-bloques;
56
57      @media screen and (max-width: $md) {
58          margin-bottom: 0;
59      }
60  }
```

Bibliografía.

A continuación, presento la relación bibliográfica que he consultado para la realización de este trabajo.

- Apuntes y recursos de la plataforma de Educación a Distancia.
- Sass y SCSS | Jairo García Rincón (jairogarciarincon.com)
- Ejercicio 3 | Jairo García Rincón (jairogarciarincon.com)
- Sass: Numeric Operators (sass-lang.com)
- :nth-of-type - CSS | MDN (mozilla.org)
- :nth-child - CSS | MDN (mozilla.org)
- Breakpoints · Bootstrap v5.0 (getbootstrap.com)