M4.252 – HTML Y CSS

PEC 3

Alumno: Pablo Andrés Antón

Índice

PRIMERA PARTE

Pregunta 1. Sobre *layouts* CSS

1. Explicad las diferencias entre los elementos *flexbox* y los elementos *grid.*

La principal diferencia entre *flexbox* y *grid* es que el primero es unidimensional, mientras que el segundo es bidimensional.

Con *flexbox* se puede trabajar en una única dimensión (horizontal o vertical). Ello no quiere decir que no se pueda trabajar en la otra dimensión, sino que no se hace simultáneamente. La propiedad "flex-direction", por ejemplo, permite cambiar de dimensión horizontal a dimensión vertical.

Con *CSS grid*, por su parte, se trabaja en dos dimensiones simultáneamente. La horizontalidad viene marcada por las filas y la verticalidad, por las columnas.

La ilustración 1 muestra un elemento que puede ser creado mediante dos códigos alternativos.

Ilustración 1. Fuente: elaboración propia.

La ilustración 2 muestra los dos códigos alternativos: *flexbox* (arriba) y *CSS grid* (abajo). Hemos omitido el código no pertinente.

```css
ul {
  border: 1px solid grey;
  width: 600px;
  list-style-type: none;
  margin: 0;
  padding: 0;
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;}

ul {
  border: 1px solid grey;
  width: 600px;
  list-style-type: none;
  margin: 0;
  padding: 0;
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-column-gap: 42px;}
```

Ilustración 2. Fuente: elaboración propia.

La disposición de los nueve rectángulos de la ilustración 1 se efectúa de manera diferente mediante *flexbox* y *CSS grid*.

Mediante *flexbox*, se utiliza la propiedad "flex-wrap" para que el contenido que sobrepase el borde se sitúe justo debajo de los tres primeros rectángulos. Así, se aprecian tres filas de tres rectángulos. Esto crea la ilusión de que se están utilizando dos dimensiones simultáneamente, cuando en realidad, se trata de una línea continua horizontal unidimensional partida en tres partes.

Mediante *grid*, se está usando la propiedad explícita "grid-template-columns", que forma tres columnas, y la propiedad implícita no declarada "grid-auto-rows", que forma tres filas. Por ello, se está trabajando de manera bidimensional. Así, se obtiene un resultado similar al de *flexbox* utilizando dos dimensiones. Aunque en el ejemplo no se utilice, existen en *CSS grid* incluso

propiedades bidimensionales en sí mismas, como “grid-gap” (o, simplemente, “gap”). Esta propiedad controla los parámetros tanto verticales como horizontales simultáneamente.

Dada la diferencia en lo que a unidimensionalidad o bidimensionalidad se refiere, se deducen otras diferencias entre ambos sistemas que se explican a continuación.

Flexbox puede resultar más sencillo y adecuado para dar forma al contenido, ya que permite dar formato en contextos unidimensionales de manera eficiente. Un menú como el de la ilustración 3, en el que solo importa el factor horizontalidad, podría ser un ejemplo.

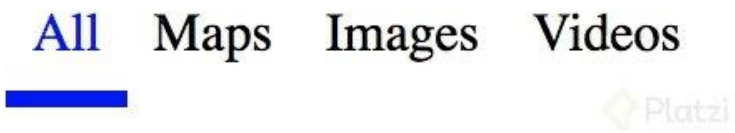

Ilustración 3. Fuente: Flexbox vs CSS Grid: ¿Cuál es la diferencia? (platzi.com).

No obstante, cuando el concepto de bidimensionalidad adquiere relevancia, *CSS grid* resulta más práctico. Por ejemplo, cuando queremos insertar el menú de la ilustración 3 dentro de la página, en una ubicación precisa respecto a los otros elementos (ilustración 4). Por ello, puede ser más recomendable *grid* para el diseño de la estructura.

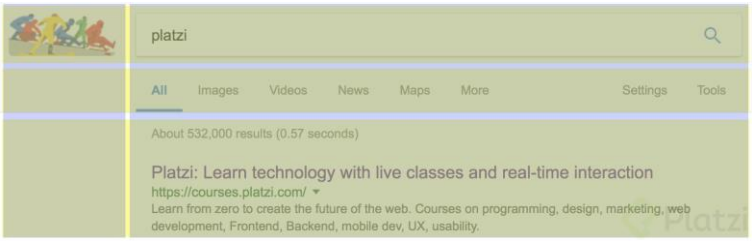

Ilustración 4. Fuente: Flexbox vs CSS Grid: ¿Cuál es la diferencia? (platzi.com).

Por todo ello, ambos no son autosuficientes, sino complementarios. Si se necesita diseñar y maquetar un sitio antes de añadir contenidos, *grid* puede ser más aconsejable. *Flexbox* es recomendable si se pretende ir añadiendo contenido para maquetarlo con posterioridad dentro de un diseño.

2. Explicad qué ventajas aporta emplear elementos *flexbox* en contraste con los métodos empleados anteriormente, como *float*.

Con el auge de dispositivos de varios tamaños, se hicieron necesarias soluciones que permitieran adaptar los sitios *web* a las características de cada dispositivo. Antes de que primero *flexbox* y luego *CSS grid layout* tomaran el relevo para esta tarea, los desarrolladores *web* pusieron en marcha algunas alternativas. Entre ellas se cuenta el uso de las propiedades "float" o "position".

Las ventajas de usar métodos modernos de puesta en papel se resumen en tres: una mayor simplicidad y rapidez, mejor adaptabilidad a los cambios, y mayor lógica, coherencia y "semántica".

Esto se explica porque el uso de *float* tenía el propósito inicial de adjuntar imágenes a columnas de texto. Sin embargo, la ausencia de métodos de composición de página eficaces como *flexbox* y *CSS grid layout* llevó a los desarrolladores *web* a pervertir el propósito inicial de *float* al extender su uso a la composición de página. Es decir, la propiedad "float" no había sido creada para ello y, por tanto, no era adecuada en términos de sencillez, lógica ni "semántica". Algo similar puede decirse de la propiedad "position" y de otros métodos.

Como norma general, usar estos u otros métodos antiguos de composición de página implica una mayor cantidad de código y de menor inteligibilidad para obtener un resultado visual similar a lo conseguido con los métodos recientes.

A la hora de llevar a cabo el mantenimiento, de efectuar cambios que implican modificar las dimensiones del contenido o los continentes, los métodos antiguos adolecen de cierta rigidez que supone un trabajo mayor para llevar a cabo las adaptaciones.

En tercer lugar, al usar los métodos modernos de composición de página, los elementos se disponen de una forma más estructurada y lógica, ya que los métodos modernos fueron concebidos para ello. Al incrementar la lógica, el resultado es más "semántico" en el sentido en que los elementos vienen a interrelacionarse de manera más "natural" a lo que se espera de ellos.

Pregunta 2. Sobre diseño responsivo

1. Explica en breves palabras qué es el diseño responsivo (*responsive design*).

El diseño responsivo hace referencia a la capacidad de un sitio *web* de "responder" a las características de la pantalla en la que se proyecta, lo que fundamentalmente (aunque no exclusivamente) se refiere al tamaño de dicha pantalla. Un diseño responsivo es aquel que sabe

lidiar con las limitaciones que una pantalla pequeña supone y aprovechar las oportunidades que otra mayor permite. De este modo, el diseño responsivo implica que un mismo sitio se adapte a diferentes pantallas, adquiriendo una apariencia diferente en función de la pantalla. Para ello no se utiliza una sola tecnología, sino un conjunto de métodos como *media queries* o métodos modernos de composición de página (*flexbox* y *Css grid*).

El diseño responsivo no solo se refiere a la cuestión del espacio, sino que debe tener en cuenta más factores, como la luminosidad ambiental a la hora de mostrar determinada paleta de colores, por ejemplo.

2. Describe detalladamente la estructura empleada en CSS para aplicar diseño responsivo.

El diseño responsivo se logra gracias a compaginar varias técnicas como *media queries* o métodos modernos de composición de página (*flexbox* y *CSS grid layout*).

Un enfoque habitual y recomendable consiste en la construcción del sitio en su versión más reducida en primer lugar, es decir, concebida para la pantalla de teléfono móvil. Es la estrategia conocida como *Mobile first*. El sitio adquiere una disposición, por lo general, en una sola columna. Este prototipo servirá de base para aplicar diversos métodos que permitirán que adquiera un diseño responsivo, es decir, que se adapte a las características de cada dispositivo.

Los métodos modernos de composición de página, fundamentalmente *flexbox* y *CSS grid layout* (aunque los *frameworks* pueden contar con sus propios métodos), permiten la creación de cuadrículas flexibles. Una cuadrícula flexible consiste en un elemento capaz de adaptarse de forma dinámica a las dimensiones del *viewport*.

En el caso de *flexbox*, hay que utilizar las propiedades “flex”, o bien “flex-basis”, “flex-grow” y “flex-schrink”.

En el caso de *CSS grid*, el uso de “fr” como unidad de medida, torna el elemento responsivo.

La tipografía se puede hacer responsiva mediante la técnica de *media queries*, que explicamos a continuación, o mediante la combinación de la unidad vw y una unidad fija (como rem) mediante “calc()”.

Todo aquello que no haya adoptado un diseño responsivo con las anteriores técnicas deberán usar *media queries*. Con esta técnica se establecen puntos de ruptura o de inflexión (*breakpoints*) que marcan a partir de qué tamaño de pantalla (fundamentalmente), determinado elemento experimenta un cambio de apariencia.

3. Explica qué podemos hacer para facilitar que la página sea *responsive* antes de dar el paso de añadir *media queries*.

En lo que se refiere a HTML, cabe utilizar las dos siguientes técnicas:

- Un metadato que colabora en el diseño responsivo es <meta name="viewport" content="width=device-width,initial-scale=1">. Esto indica al navegador del dispositivo móvil que equipare la anchura del *viewport* a la anchura de la pantalla, evitando que el navegador del móvil use una anchura mayor.

- Para el caso de las imágenes, se utiliza el elemento "<picture>" con los atributos de "<img>" "srcset" y "sizes" acompañado de metadatos que dan indicaciones al navegador para que escoja entre una u otra imagen de determinado tamaño o determinadas características preseleccionada para la pantalla en cuestión.

En cuanto a CSS, tal y como se ha detallado en la pregunta anterior, el uso de los llamados métodos modernos de composición de página (y *CSS grid layout*), así como de la combinación de la unidad vw y una unidad fija (como rem) mediante "calc()", facilitan que la página sea responsiva antes de proceder a usar *media queries*.

SEGUNDA PARTE

El sitio *web* ha sido alojado en Github Pages y es accesible mediante el enlace siguiente:

https://p9andres.github.io/Oscarsclub-Pablo-Andres/.

Pregunta 4. Explicación de las entidades HTML utilizadas.

Cada página se ha dividido en dos partes: cabecera y cuerpo. El cuerpo, a su vez, se ha subdividido en tres partes.

- '<header>' para el encabezado ('<nav>', para el menú de navegación, se encuentra dentro de este),
- '<main>' para el contenido principal (que incluye varias '<section>')
- y '<footer>' para el pie de página.

Primero se expondrá la cabecera y después, los dos elementos comunes a los cuatro archivos .html (<header> y <footer>). Por último, el elemento '<main>', que difiere en cada una de las cuatro páginas así como el hilo de Ariadna, que no forma parte de '<main>'.

Antes de proceder a exponer las entidades de la cabecera, conviene recordar que el elemento '<html>' engloba cabecera y cuerpo. A este se la ha asignado un atributo 'lang' para el idioma. También se ha utilizado '<!DOCTYPE html>', requisito indispensable para poder escribir un archivo de formato .html.

Cabecera (<head>)

Una serie de elementos '<meta>' a los que se asocia una serie de atributos, definen algunos metadatos: autor, descripción y configuración tipográfica (charset="utf-8"). Se incluye también el título de la página (<title>), el enlace al archivo de CSS (<link rel="stylesheet" type="text/css" href="css/styles.css">) y el enlace al archivo normalize.css para armonizar el renderizado del sitio en los distintos navegadores. Por último, se encuentra los enlaces al tipo de fuente y el elemento <meta name="viewport" content="width=device-width,initial-scale=1">. Este último contribuye al diseño responsivo, como hemos explicado arriba.

Encabezado (<header>)

Dentro de '<header>' se encuentra dos elementos ' ' y un menú de navegación '<nav>':

- El primer ' ' alberga un botón para iniciar sesión.
- El segundo ' ' contiene un enlace (<a>), que a su vez alberga el logotipo (<img>). Lleva un atributo 'src' que enlaza al archivo y 'alt', para mostrar texto alternativo en caso de que la imagen no se cargara o tardara en hacerlo. Además, se encuentra el título del sitio (<h1>) con un elemento '' para permitir cambiar el estilo de la segunda palabra.
- A continuación, se sitúa un menú de navegación '<nav>'. Incluye una lista desordenada '<ul>' con dos elementos '<li>' que, a su vez, anidan un enlace a las otras páginas del sitio (excepto cuando nos encontramos en dicha página).

Pie de página (<footer>)

Incluye dos elementos ' ' y un enlace (<a>):

- El primer ' ' contiene dos menús de navegación (<nav>) con una lista desordenada '<ul>' con tres elementos '<li>' cada uno, que a su vez, anidan un enlace. Los enlaces se encuentran vacíos.
- El segundo ' ' contiene tres párrafos. Los dos últimos tienen enlaces '<a href="">' hacia sitios externos. Por ello, contienen los atributos 'title' y 'target'.
- El enlace (<a>) sirve para volver a la parte superior de la página.

Contenido principal (<main>) de 'index.html'

Tiene tres secciones (<section>):

- La primera sección contiene un título '<h2>' (<h1> se reserva para el logotipo) y un párrafo.
- La segunda sección contiene un título '<h2>' y un ' '. Este alberga cuatro elementos '<figure>'. En cada uno de ello se incrustan una imagen (con atributos para anchura, fuente y texto alternativo) y su título (<figcaption>).
- La tercera sección contiene también un título '<h2>' y un ' '. Este último alberga otro ' ' con párrafos y un formulario (<form>). El formulario contiene varios '<input>' de distintos tipos (atributo 'type') con varios atributos ('placeholder', para el

texto en el interior del campo; ‘pattern’, para el patrón de validación, etc.). También hay elementos ‘’ para resaltar palabras importantes y un botón tipo *submit* que envía el formulario.

Contenido principal (<main>) de ‘clubes-tematicos.html’

Los elementos utilizados son muy similares a los ya explicados. En este caso, el contenido se entructura en una sección que anida seis ‘ ’, cada uno de los cuales contiene un elemento ‘<figure>’ con ‘<figcaption>’ y también un párrafo y un enlace.

Contenido principal (<main>) de ‘sesiones-2020.html’

De nuevo se utilizan elementos muy similares a los explicados arriba. A lo largo de la página se citan múltiples obras cinematográficas, para cuyos títulos se emplea el elemento ‘<cite>’.

Contenido principal (<main>) de ‘detalle-sesion.html’ e hilo de Ariadna

La principal innovación consiste en un hilo de Ariadna situado entre el encabezado y el contenido principal. El hilo de Ariadna usa la estructura de un menú de navegación como los de arriba.

El contenido principal se estructura en un ‘<h2>’ y un ‘ ’. El segundo contiene un elemento ‘<aside>’ y otro ‘ ’:

- El elemento ‘<aside>’ destaca por la lista de definiciones (<dl>) albergada, con elementos ‘<dt>’ y ‘<dd>’. En estos, se ha usado ‘<b>’ por la ausencia de semántica en la negrita.
- El ‘ ’, alberga tres secciones. La última de ellas contiene un elemento ‘<iframe>’ para dar cabida a un enlace a un vídeo externo. Los atributos que lleva aparejados se han obtenido de la página del sitio externo.

Pregunta 5. Explicación de las entidades CSS utilizadas.

Hemos optimizado el modelo móvil para una resolución de pantalla de 360x640. Además del archivo ‘style.css’, cuyo código explicamos a continuación, hemos utilizado un archivo genérico ‘normalize.css’, cuya explicación no hemos detallado.

```
/*GENERAL*/

body {
    margin: 0;
} Elimina el margen, para que el pie de página ocupe el 100% del ancho de página.

header {
    margin-bottom: 50px;
} Al aumentar el margen inferior del encabezado, se imita la distancia del
modelo.

main {
    margin: 1rem;
} Permite que los elementos no toquen el borde de la pantalla.

section {
    margin-top: 50px;
    margin-bottom: 50px;
} Se intenta replicar los márgenes del modelo.

footer {
    margin-top: 80px;
    padding: 1rem;
    text-align: right;
} Margen y relleno tratan de emular el original. El símbolo de “subir” se
desplaza a la derecha mediante la alineación de texto.



/*FONT*/

html {
    font-size: 12px;
    line-height: 1.7;
} Se reduce el tamaño de fuente y se aumenta el interlineado, a imagen del
modelo.

button {
```

```
    font-size: 1rem;
} Tamaño de fuente a imagen del modelo en el botón.

h2, h3, .unete div p {
    font-size: 1.5rem;;
} Tamaño de fuente a imagen del modelo.

@media (min-width: 850px) {
    h2 {
        font-size: 1.8rem;
    }
} A partir de una anchura de 850px, se aumenta el tamaño de fuente de ‘h2’
Mediante media queries.

figcaption {
    font-size: 1.2rem;
} Tamaño de fuente a imagen del modelo.

.unete form p {
    font-size: 0.6rem;
} Tamaño de fuente a imagen del modelo.

footer a[href="#"] {
    font-size: 2rem;
} Tamaño de fuente a imagen del modelo.

.div2-footer {
    font-size: 0.9rem;
    line-height: 1;
} Tamaño de fuente e interlineado a imagen del modelo.

.menu-italics {
    font-style: italic;
} Se transforma en cursiva el botón del menú de la página actual.

.section-carteles a {
    font-weight: 700;
} Se pasa a negrita.

.section-carteles ul a {
    color: initial;
    font-size: 0.9rem;
} Se vuelve al color negro original y se reduce el tamaño.

body, h1 {
```

```
    font-family: 'Noto Serif', serif;
} Establece este tipo de fuente para todo el contenido y ‘h1’.

header, footer, form, .ariadna, button, h4, .section-carteles li {
    font-family: 'Open Sans', sans-serif;
} Se revierte la orden anterior y se cambia el tipo de fuente para todos estos
elementos.

.obra-autor-trailer figcaption {
    font-variant: small-caps;
} Se usan versalitas para este elemento.


/*COLORS*/

body {
    background-color: rgb(232, 232, 232);
    color: rgb(28, 28, 28);
} Color de fondo y color de texto general.

footer {
    background-color: rgb(67, 58, 68);
    color: rgb(255, 255, 255);
} Color de fondo y color de texto del pie de página.

button {
    background-color: rgb(155, 23, 0);
    color: rgb(255, 255, 255);
    border: rgb(155, 23, 0);
} Colores de los botones: fondo, texto y borde.

button:focus {
    background-color: rgb(28, 28, 28);
} Color de fondo en estado focus. Lo hemos incluido para mejorar la
accesibilidad.

button:hover {
    background-color: rgb(28, 28, 28);
} Color de fondo en estado hover.

button:active {
    background-color: green;
} Color de fondo en estado active. Lo hemos incluido para hacer más evidente la
acción.
```

```
/*LINKS*/

a {
    text-decoration: none;
} Elimina el subrayado de todos los enlaces.

a:link {
    color: rgb(28, 28, 28);
} Color de texto de los enlaces.

a:visited {
    color: rgb(28, 28, 28);
} Color de texto de los enlaces visitados. No se especificaba, así que hemos
optado por no hacer distinción.

a:focus {
    opacity: 0.5;
} Los enlaces en estado focus adquieren transparencia.

a:hover {
    opacity: 0.5;
} Los enlaces en estado hover adquieren transparencia.

a:active {
    color: green;
} Los enlaces en estado activo se vuelven verdes para hacer más evidente la
acción.

.main a {
    color: rgb(155, 23, 0);
} Color de los enlaces de 'main', distinto al color general.

.main a:focus {
    color: rgba(155, 23, 0, 0.5);
} Color de los enlaces de 'main' en estado focus.

.main a:hover {
    color: rgba(155, 23, 0, 0.5);
} Color de los enlaces de 'main' en estado hover.

.main a:active {
    color: green;
} Color de los enlaces de 'main' en estado activo.

.footer a {
```

```
    color:rgb(255, 255, 255);
} Color de los enlaces del pie de página.

.footer a:focus {
    color: rgb(155, 23, 0);
} Color de los enlaces del pie de página en estado focus.

.footer a:hover {
    color: rgb(155, 23, 0);
    opacity: 1;
} Color de los enlaces del pie de página en estado hover.

.footer a:active {
    color: green;
} Color de los enlaces del pie de página en estado activo.



/*HEADER*/

.div-button {
    margin: 0.5rem 1rem;
} Imita los márgenes del original.

button {
    padding: 0.4rem 0.8rem;
    border-radius: 0.3em;
} Imita el diseño original del botón: relleno y bordes redondeados.

.div-isologo {
    margin: 0 0.5rem 1.2rem;
} Imita los márgenes del original.

.div-isologo img {
    position: relative;
        top: 15px;
} Desplaza la imagen hacia abajo, para que la claqueta se alinee con las letras.

h1 {
    display: inline-block;
    margin: 0;
} Permite que el texto se encuentre en la misma línea que la claqueta y se sitúe
yuxtapuesta.

span {
```

```
    color: rgb(155, 23, 0);
    font-style: italic;
} permite volver roja y cursiva la segunda palabra.

header li {
    list-style-type: none;
    font-weight: 700;
    padding: 0 1rem;
} Elimina las viñetas, transforma en negrita y adapta el relleno.

header ul {
    padding: 0;
    margin: 0.5rem 0;
    display: flex;
    flex-direction: column;
    align-items: center;
} Dispone el menú en posición vertical y centrada para optimizarlo para pantallas
estrechas.

header {
    display: flex;
    flex-direction: column;
    align-items: center;
    min-width: 196px;
} Dispone el menú en posición vertical y centrada para optimizarlo para pantallas
estrechas. Establece una anchura mínima para que conserve la forma si se reduce
más.

@media (min-width: 600px) {
    header {
        flex-direction: row;
        justify-content: space-between;
    } A partir de 600px el encabezado se dispone horizontalmente, con los
elementos alejados entre sí.
    .div-button {
        order: 3;
    } El botón pasará a la derecha del todo.
    header ul {
        flex-direction: row;
        justify-content: space-between;
        align-items: center;
    } A partir de 600px el encabezado se dispone horizontalmente, con los
elementos alejados entre sí.
}
```

```
/*FOOTER*/

footer li {
    list-style-type: none;
} Se eliminan las viñetas.

footer ul {
    padding-left: 0;
} Se elimina el relleno izquierdo original para poder centrar los elementos.

footer li, footer p {
    display: flex;
    flex-direction: column;
    align-items: center;
} Disposición vertical y centrada, optimizada para pantallas estrechas.

.div1-footer nav {
    margin: 40px 0;
} Se establece márgenes superior e inferior como en el modelo.

.div1-footer nav {
    margin-bottom: 30px;
} Se establece margen inferior como en el modelo.

@media (min-width: 600px) {
    .div1-footer {
        display: flex;
        flex-direction: row;
        justify-content: space-between;
    } A partir de 600px, la disposición horizontal adapta la web a pantallas
anchas.
    footer ul {
        display: flex;
        flex-direction: row;
        justify-content: space-between;
    } A partir de 600px, la disposición horizontal adapta la web a pantallas
anchas.
    .div2-footer p {
        align-items: flex-start;
    } Este elemento se situará a la izquierda.
    .vinneta::after {
```

```
    content: " • ";
    margin: 0 0.9rem;
    }Se añade viñetas.
}



/*INDEX section ¿Qué es Oscars?*/

.que-es-oscars {
    display: flex;
    flex-direction: column;
    text-align: center;
} Disposición vertical y centrada, optimizada para pantallas estrechas.

.que-es-oscars p {
    max-width: 530px;
    margin-left: auto;
    margin-right: auto;
} Máximo de anchura y márgenes automáticos, para centrar el elemento.

/*INDEX section ¿Cómo funciona?*/

h2 {
    text-align: center;
} Centrado del elemento.

.como-funciona div {
    display: flex;
    flex-direction: column;
    text-align: center;
} Disposición vertical y centrada, optimizada para pantallas estrechas.

@media (min-width: 500px) {
    .como-funciona div {
        flex-direction: row;
        flex-wrap: wrap;
        justify-content: center;
    } A partir de 500px la disposición horizontal se adapta a pantallas más
anchas.
}

/*INDEX section Únete al club*/

.div-unete div {
```

```
    display: flex;
    flex-direction: column;
    min-width: 336px;
} Disposición vertical y centrada, optimizada para pantallas estrechas.

.div-unete p:nth-child(1), .div-unete p:nth-child(2) {
    margin-top: 0;
} Trata de imitar el espaciado del modelo eliminando el margen superior.

.div-unete p:nth-child(1) {
    margin-bottom: 0;
} Ídem.

.div-unete form {
    display: flex;
    flex-direction: column;
    min-width: 336px;
    max-width: 336px;
} Disposición vertical y centrada, optimizada para pantallas estrechas.

.div-unete button {
    width: 110px;
    height: 34px;
    margin-top: 0.5rem;
} Cambia dimensiones del botón.

.div-unete {
    display: flex;
    flex-direction: column;
    align-items: center;
} Disposición vertical y centrada, optimizada para pantallas estrechas.

@media (min-width: 750px) {
    .div-unete {
        flex-direction: row;
        justify-content: center;
        align-items: flex-start;
    } A partir de 750px, la disposición horizontal se adecua a pantallas más
anchas.
    .div-unete div {
        max-width: 336px;
    } Anchura máxima.
}

@media (min-width: 850px) {
```

```
    .div-unete div, .div-unete form {
        margin: 0 75px;
    } Cambio en el margen para emular el modelo.
}

.unete input[type="text"], .unete input[type="email"], .unete input[type="passwor
d"]  {
    max-width: 336px;
    height: 30px;
    margin-top: 0.5rem;
} Cambio de dimensiones de los elementos 'input'.

.unete input:valid {
    border: solid green 2px;
} Borde verde para validar el formulario.



/*CLUBES TEMÁTICOS*/

figure {
    margin: 2rem auto;
} El borde automático centra la figura.

.div-clubes figcaption::before {
    content: " ...... ";
    margin: 0 0.5rem 0 0.1rem;
    color: rgb(155, 23, 0);
} Se añade puntos rojos.

.div-clubes div {
    max-width: 336px;
} Se establece anchura máxima.

.div-clubes {
    display: flex;
    flex-direction: column;
    max-width: 1075px;
    margin-left: auto;
    margin-right: auto;
} Disposición vertical y centrada, con anchura máxima.

@media (min-width: 500px) {
    .div-clubes {
        flex-direction: row;
```

```
        flex-wrap: wrap;
        justify-content: center;
    } Disposición horizontal y centrada, para pantallas a partir de 500px. 'Flex-
wrap' permite que una línea se sitúe debajo de la otra.
}



/*SESIONES 2020*/

.sesiones-2020 {
    max-width: 600px;
    margin-left: auto;
    margin-right: auto;
} Establece una anchura máxima y centra el contenido.

.section-carteles ul {
    padding-left: 0;
    list-style-type: none;
} Elimina las viñetas y el relleno izquierdo.

h3 {
    margin-top: 1rem;
    margin-bottom: 1rem;
} Cambia el margen superior e inferior.

.div-carteles figure {
    background-color: rgb(255, 255, 255);
    width: 180px;
    height: 310px;
    margin-top: 0;
    margin-bottom: 0;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
} Establece un fondo blanco de determinadas dimensiones. Además dispone los dos
elemtos existentes en los extremos superior e inferior.

#de-otra-epoca figure {
    height: 340px;
} Cambia la altura.

.div-carteles div {
    display: flex;
    flex-direction: column;
```

```
    max-width: 600px;
    margin-left: auto;
    margin-right: auto;
} Establece la anchura máxima, la disposición vertical y el centrado.

.div-carteles h4::before {
    content: " ··· ";
    margin: 0 0.5rem 0 1rem;
    color: rgb(155, 23, 0);
} Añade tres puntos rojos.

.div-carteles h4 {
    margin-bottom: 0.5rem;
} Cambia el tamaño del margen inferior.

@media (min-width: 550px) {
    .div-carteles div {
        flex-direction: row;
        justify-content: center;
    } A partir de 550px, la disposición se vuelve horizontal y el contenido se
centra.
    .div-carteles section {
        max-width: 600px;
        margin-left: auto;
        margin-right: auto;
    } La anchura queda limitada a 600px y el contenido se centra.
    .section-carteles {
        border-top: rgb(155, 23, 0) solid 1px;
    } Se crea un borde en la parte superior.
}

.section-carteles {
    max-width: 790px;
    margin-left: auto;
    margin-right: auto;
} La anchura queda limitada y el contenido se centra.

@media (min-width: 800px) {
    .section-carteles {
        display: flex;
        justify-content: space-between;
    } A partir de 800px, los elementos se distribuyen dejando los espacios entre
sí.
    .div-carteles figure {
        margin-left: 10px;
```

```
        margin-right: 10px;
    } Se cambia el margen izquierdo y derecho.
}



/*DETALLE-SESIÓN*/

.ariadna li:last-of-type {
    color: rgba(28, 28, 28, 0.5);
} Se cambia el color del último elemento del hilo de Ariadna.

.ariadna li:last-of-type::before, .ariadna li:nth-of-type(2)::before  {
    content: " / ";
    margin: 0 5px;
} Se añaden barras al hilo.

.ariadna li {
    display: inline-block;
} Se dispone horizontalmente.

.ariadna ul {
    padding-left: 0;
} Se elimina el relleno izquierdo.

.ariadna {
    text-align: center;
    max-width: 800px;
    margin-left: auto;
    margin-right: auto;
    padding: 0 1rem;
} Se centra el contenido. Se establece una anchura máxima.

@media (min-width: 600px) {
    .ariadna ul {
        text-align: left;
    } A partir de 600px, el contenido se desplaza a la izquierda.
}

.h2-klaus {
    text-align: left;
    max-width: 800px;
    margin-left: auto;
    margin-right: auto;
} Se centra el contenido. Se establece una anchura máxima.
```

```
.div-klaus {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    max-width: 850px;
    margin-left: auto;
    margin-right: auto;
} Se dispone el contenido horizontalmente, pero al estrecharse, la propiedad 'flex-wrap' simulará una disposición vertical. El contenido queda centrado y con una anchura máxima.

@media (min-width: 800px) {
    .obra-autor-trailer {
        margin-left: 2rem;
    } A partir de 800px, se crea un margen para evitar que los elementos de flexbox se lleguen a juntar.
}

aside {
    max-width: 336px;
    margin-bottom: 50px;
} Se establece una anchura máxima y se aumenta el margen inferior para cuando se visione en pantallas estrechas.

dt {
    font-weight: 700;
    border-bottom: solid black 1px;
} Se crea una línea a modo de subrayado y se pasa a negrita.

dl {
    margin-bottom: 0;
} Se elimina el margen para imitar la apariencia del modelo.

.obra-autor-trailer {
    max-width: 450px;
    margin-top: 0;
} Se establece una anchura máxima y se elimina el margen superior.

.div-klaus h3 {
    margin-top: 0;
} Se elimina el margen superior.

.obra-autor-trailer img {
    border-radius: 50%;
```

```
    float: left;
    shape-outside: circle();
    width: 110px;
} Se convierte la imagen en un círculo que se inscribe en el vértice izquierdo
superior del texto. El texto a la derecha del círculo adopta una forma cóncava.
Se cambia el tamaño.

@media (min-width: 474px) {
    iframe {
        width: 450px;
        height: 253px;
    } A partir de 474px, se aumenta el tamaño.

}

.obra-autor-trailer section:first-of-type  {
    margin-top: 0;
} Se elimina el margen superior.

.obra-autor-trailer, .obra-autor-trailer section:last-of-type, .obra-autor-
trailer section:last-of-type figure {
    margin-bottom: 0;
} Contribuye a reducir la separación respecto al pie de página.
```

BIBLIOGRAFÍA

The MDN web docs [internet]. Mozilla; 2020 [consultado el 27 de noviembre de 2020]. Disponible en: MDN Web Docs (mozilla.org).

W3C (MIT, ERCIM, KEIO, BEIHANG). W3C Markup Validator [internet]. V1.3+hg. 1994-2013. Disponible en: The W3C Markup Validation Service.

W3C (MIT, ERCIM, KEIO, BEIHANG). W3C Markup Validator [internet]. 1994-2009. Disponible en: The W3C CSS Validation Service.

MIT y Microsoft Corporation. Visual Studio Code [software]. Versión 1.51.1. 2020.

W3.CSS. W3schools [Internet]. W3.CSS; 1999-2021[consultado el 26 de diciembre de 2020]. Disponible en: HTML pattern Attribute (w3schools.com)

Webgenio. Webgenio [Internet]. Webgenio [consultado el 26 de diciembre de 2020]. Disponible en: Diferencias entre CSS Grid y Flexbox, cuál utilizar – WebGenio

Neal, J. y Gallagher, N. Github [Internet]. Normalize.css. Versión 11.0.1. Agosto de 2020 [consultado el 26 de diciembre de 2020]. Disponible en: normalize.css (csstools.github.io)