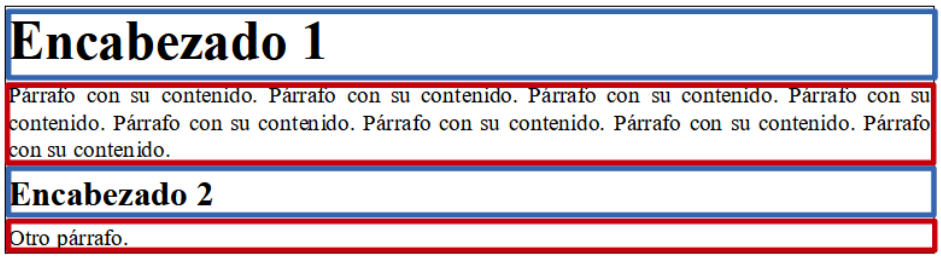
Posicionamiento

En este documento vamos a ver qué propiedades tienen los elementos CSS para definir su posicionamiento en las páginas y distancia respecto al resto, así como las estrategias que podemos aplicar para establecer la ubicación o posición de los distintos componentes de la página.

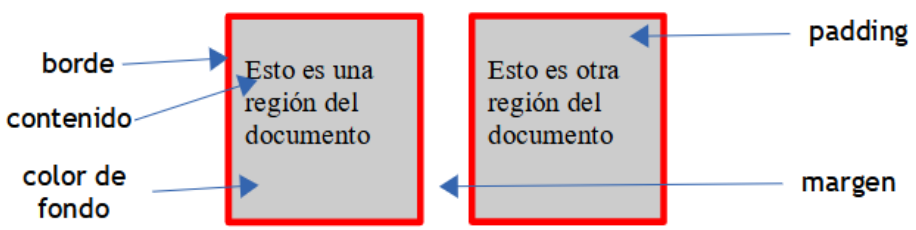
1. El modelo de cajas CSS

El modelo de cajas en CSS es uno de los aspectos más complicados de dominar, pero que más versatilidad puede dar a nuestras webs. Este modelo se basa en que cada elemento de un documento HTML (cada párrafo, *h1*, *div*, etc, incluso elementos en línea como negritas o *spans*) tiene una caja, un rectángulo que lo engloba y lo separa del resto.



Así, CSS permite definir las características de cada una de estas cajas (altura, anchura, posición, color...). Para ello, hemos de tener en cuenta que cada caja se compone de ciertos elementos:

- El **contenido** HTML que tenga
- El **borde**, un recuadro (visible o no) que encierra al contenido
- El **relleno o *padding***, que es el espacio entre el contenido y el borde
- El **margen (margin)**, que es el espacio entre la caja y otras cajas adyacentes.
- El tamaño de la caja (**anchura y altura**)
- El **color de fondo** del elemento. Alternativamente, también se puede definir una imagen de fondo con la propiedad *background-image*, que se superpone al color de fondo.



2. Cambiando algunas propiedades básicas

Veamos a continuación cómo definir ciertas propiedades básicas de las vistas anteriormente, tales como márgenes, rellenos, bordes, etc.

2.1. Definiendo los márgenes

Para definir márgenes, podemos usar la propiedad `margin` y definir separados por espacios los márgenes de los 4 lados, en el orden *arriba*, *derecha*, *abajo*, *izquierda*:

```
p
{
    margin: 10px 5px 2px 5px;
}
```

Alternativamente, también podemos usar las propiedades `margin-top`, `margin-right`, `margin-bottom` y `margin-left` y definir cada margen por separado. Estas regla sería equivalente a la anterior:

```
p
{
    margin-top: 10px;
    margin-right: 5px;
    margin-bottom: 2px;
    margin-left: 5px;
}
```

Si en la propiedad `margin` usamos un solo valor, los 4 márgenes toman ese mismo valor. Si usamos dos valores, el primero se aplica arriba y abajo, y el segundo a izquierda y derecha. Si usamos tres valores, el primero es para arriba, el segundo para izquierda y derecha, y el tercero para abajo.

2.2. Definiendo el relleno o *padding*

Del mismo modo, podemos emplear la propiedad `padding` para definir la distancia de relleno interior del elemento (usando los mismos valores y en el mismo orden que *margin*), y también podemos optar por especificar los cuatro rellenos por separado con `padding-top`, `padding-bottom`, `padding-left` y `padding-right`.

```
div
{
    padding: 10px 20px 5px 20px;
}
```

2.3. Definiendo los bordes

Para el borde, podemos modificar varias propiedades del mismo: su estilo (simple, doble, punteado, etc.), su color, su grosor, su radio... Para ello, tenemos las propiedades `border-style`, `border-color`, `border-width` y `border-radius`, respectivamente.

Entre los estilos más utilizados está el simple continuo (*solid*), el doble (*double*), el punteado (*dotted*) o ninguno (*none*, valor por defecto). Por ejemplo, para un borde de un *h1* de color rojo y trazo doble, con grosor de 5 píxeles, pondríamos:

```
h1 { border-style: double; border-color: red; border-width: 5px; }
```

Podemos definir estilos de borde para cada lado, colocando el lado en medio de la propiedad. Por ejemplo, para modificar el color del borde inferior, pondríamos `border-bottom-color`, y para cambiar el grosor del borde derecho, pondríamos `border-right-width`. Así, el siguiente ejemplo establece un borde inferior y derecho simple, de color azul y 2 píxeles de grosor:

```
h1
{
    border-bottom-style: solid; border-right-style: solid;
    border-bottom-color: blue; border-right-color: blue;
    border-bottom-width: 2px; border-right-width: 2px;
}
```

También podemos usar las propiedades `border-top`, `border-bottom`, `border-left` y/o `border-right` (e incluso la propiedad `border` a secas) y definirles sus tres propiedades (estilo, grosor y color) separadas por espacios:

```
h1 { border-bottom: solid 2px blue; }
```

En lo que respecta al radio del borde, desde CSS3 se permite definir bordes redondeados dotándoles de un radio de circunferencia. Un valor de 0 dejaría un borde cuadrado, y cuanto mayor valor le demos, más redondeado será dicho borde. Podemos aplicar el radio a todos los bordes del elemento, o a alguno(s) en concreto:

```
/* Borde superior izq, superior der, inferior izq, inferior der */
h1 { border-radius: 5px 5px 10px 10px; }

/* Mismo radio en todos los bordes */
p { border-radius: 10px; }
```

2.4. Definiendo el fondo

Ya hemos visto anteriormente que la propiedad `background-color` define el color de fondo del elemento sobre el que se aplica, y que dicho color lo podemos expresar como un color simple en inglés o en formato RGB (decimal o hexadecimal):

```
p { background-color: ##12FF3A; }
```

Adicionalmente, podemos indicar una imagen de fondo para el elemento mediante la propiedad `background-image`. En este caso indicamos la ruta relativa desde la carpeta principal de la aplicación web:

```
div { background-image: url("imagenes/foto.png"); }
```

Esta imagen se pondrá por encima del color de fondo que haya definido, si hay alguno. Si la imagen no es lo suficientemente grande como para abarcar todo el elemento, podemos indicar que se repita vertical u horizontalmente añadiendo la propiedad `background-repeat`, que puede valer *repeat-x* (repetir horizontalmente), *repeat-y* (repetir verticalmente), *repeat* (repetir en todas direcciones) o *no-repeat* (no repetir, valor por defecto)

```
div
{
    background-image: url("imagenes/foto.png");
    background-repeat: repeat-x:
}
```

2.5. Definiendo el tamaño

Finalmente, podemos cambiar el tamaño de la caja modificando sus dimensiones (anchura y altura) con las propiedades `width` y `height`, utilizando el tipo de medida que queramos (normalmente, porcentajes o píxeles).

```
h1 { width: 70%; height: 50px; }
```

También existen las propiedades `max-width`, `min-width`, `max-height` y `min-height` para establecer unos rangos de anchura y altura variables.

> **NOTA IMPORTANTE**: cuando definimos el tamaño (*width* y *height*) de un elemento, este tamaño se define sin contar el tamaño del posible borde ni del *padding*. Si añadimos después estas propiedades al elemento, las dimensiones de las mismas se añaden al tamaño inicial.

> **Ejercicio 1:**
>
> Crea un documento llamado **cajasCSS.html** que, a través de un documento CSS externo, defina un selector de clase llamado *caja*, con los siguientes estilos para una caja:
>
> - Margen izquierdo y derecho del 20%, superior de 30 píxeles e inferior de 0
> - Relleno (*padding*) de 10 píxeles a cada lado
> - Borde superior rojo de 4 píxeles de grosor, derecho azul de 6 píxeles e inferior verde de 2 píxeles, todos con estilo sólido.
> - Color de fondo amarillo
>
> Finalmente, crea un párrafo (más o menos largo, de unas 3 o 4 líneas al menos) y aplícale este estilo.

3. Posicionando los elementos

Existen diferentes alternativas para definir la posición de un elemento en una página. Aquí vamos a analizar dos de ellas. La primera viene incorporada desde CSS2, aunque ya está más en desuso, y consiste en definir el tipo de posicionamiento de cada elemento, junto con su tamaño (anchura y/o altura) para que se ubique respecto al resto. La segunda alternativa, más reciente, consiste en tratar la página como una rejilla (*grid*) de casillas, y definir qué casillas queremos que ocupe cada elemento.

3.1. Posicionamiento CSS2

A la hora de colocar cada una de las cajas (elementos HTML) en la página, el navegador sigue una serie de pautas, según las características de cada caja y el estilo que tenga configurado en el CSS. Existen diferentes tipos de posicionamiento:

- **Normal**: el que utiliza el navegador por defecto, si no se le indica lo contrario en el CSS. Depende de las características de cada caja (si es un elemento en línea o de bloque, si su tamaño permite tener otro elemento al lado, el posicionamiento que tiene el elemento que lo contiene, etc.)
- **Relativo**: se desplaza la caja respecto a lo que sería su posicionamiento normal
- **Absoluto**: la caja se sitúa en una posición absoluta respecto de su elemento contenedor
- **Fijo**: la caja se sitúa en una posición fija en pantalla, independientemente de si el usuario sube o baja la página.
- **Flotante**: la caja se sitúa todo lo posible a la izquierda o derecha, dentro de la línea horizontal en que se encuentra.

El tipo de posicionamiento se establece con la propiedad `position`, que puede valer cualquiera de los tipos anteriores, salvo el flotante: `static` (normal), `relative`, `absolute` o `fixed`. Por ejemplo, la siguiente caja tiene un posicionamiento relativo 20 píxeles por debajo y 10 píxeles más a la derecha de su posicionamiento normal:

```
div { position: relative; top: 20px; left: 10px }
```

En el caso del posicionamiento flotante, se especifica con la propiedad `float`, que puede valer `left`, `right` o `none`.

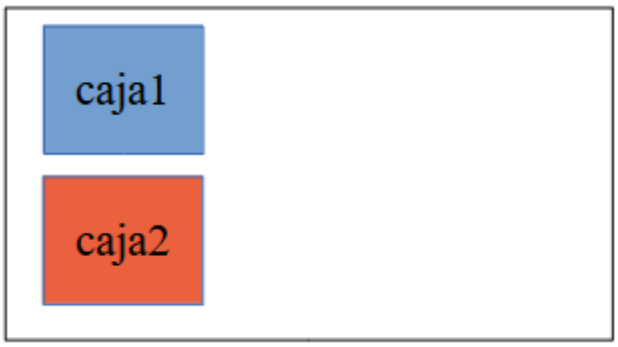
- Si no existe ningún tipo de posicionamiento flotante, las cajas se posicionan según su configuración natural. Así, los elementos de bloque, como *h1*, *p*, *div*, etc., se posicionan uno debajo del otro.
- Si algún elemento tiene posicionamiento flotante a izquierda o derecha, queda alineado a la izquierda o derecha de la línea que ocupa. Si son varios los que tienen ese mismo posicionamiento, quedan colocados uno junto a otro, dependiendo de la anchura que ocupe cada uno, hasta completar la línea (se continúa en la siguiente línea si no hay más espacio).

Veamos algunos ejemplos:

Caso 1: sin posicionamiento

En este caso no se indica posicionamiento en ninguna de las cajas, por lo que se colocan una debajo de la otra, aunque quepan juntas en la misma línea, porque son elementos en bloque (*div*).

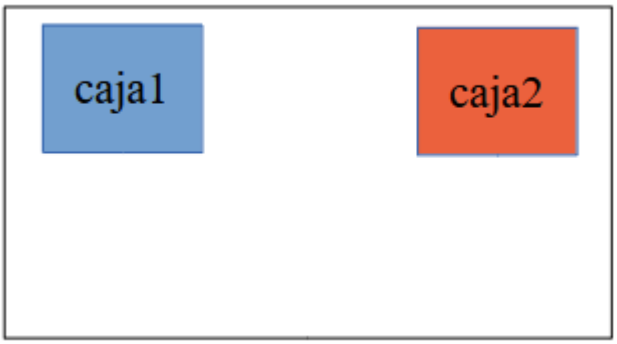
```
#caja1
{
    background-color: blue;
    width: 30%;
    height: 100px;
    text-align:center;
    margin: 20px;
}

#caja2
{
    background-color: red;
    width: 30%;
    height: 100px;
    text-align:center;
    margin: 20px;
}
```



Caso 2: posicionamiento izquierda y derecha

En este caso la *caja2* tiene un posicionamiento flotante a la derecha y la *caja1* a la izquierda.

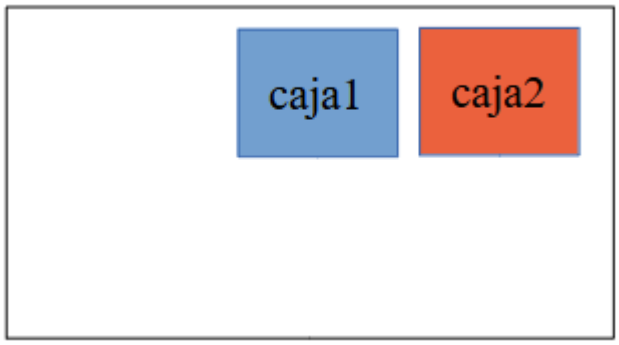
```
#caja1
{
    background-color: blue;
    width: 30%;
    height: 100px;
    text-align:center;
    margin: 20px;
    float: left;
}

#caja2
{
    background-color: red;
    width: 30%;
    height: 100px;
    text-align:center;
    margin: 20px;
    float: right;
}
```



Caso 3: posicionamiento derecha

Ahora ambas cajas tienen posicionamiento a la derecha, por lo que, si caben, aparecen una junto a otra. El orden en que aparecen dependerá del orden en que se añadan en el documento HTML. En este ejemplo, se habría añadido primero la *caja2* y luego la *caja1*.

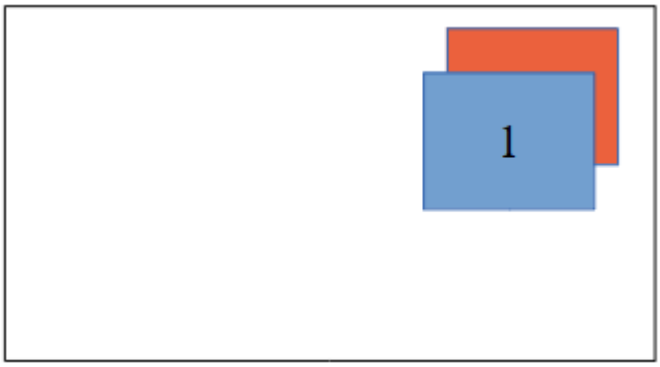
```
#caja1
{
    background-color: blue;
    width: 30%;
    height: 100px;
    text-align:center;
    margin: 20px;
    float: right;
}

#caja2
{
    background-color: red;
    width: 30%;
    height: 100px;
    text-align:center;
    margin: 20px;
    float: right;
}
```



Caso 4: posicionamiento fijo

En este caso la *caja1* tiene un posicionamiento fijo, a 50px del borde superior y otros 50px del borde derecho. Si primero colocamos la *caja2* con posicionamiento flotante derecha y luego la *caja1*, ésta se superpondrá parcialmente a la *caja2*, al compartir un espacio común por su posicionamiento fijo.

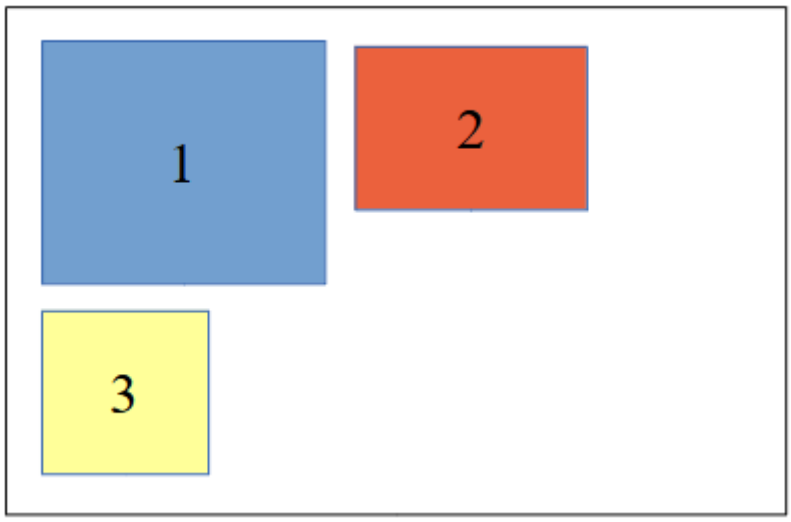
```
#caja1
{
    background-color: blue;
    width: 30%;
    height: 100px;
    text-align:center;
    margin: 20px;
    position: fixed;
    top: 50px;
    right: 50px;
}

#caja2
{
    background-color: red;
    width: 30%;
    height: 100px;
    text-align:center;
    margin: 20px;
    float: right;
}
```



Caso 5: posicionamiento izquierdo excluyente

En el caso de que necesitemos que una caja quede pegada al borde izquierdo o derecho y que ninguna caja le "moleste" para colocarse allí, podemos limpiar la línea con la propiedad `clear`, que puede valer `left`, `right` o `both`, dependiendo de a qué lado queramos ajustar las cajas. Por ejemplo, en el siguiente caso, si queremos que la caja 3 quede ajustada al margen izquierdo, le pondríamos en su estilo:

```
.caja1 { float: left; }
.caja2 { float: left; }
.caja3 { float: left; clear:left }
```

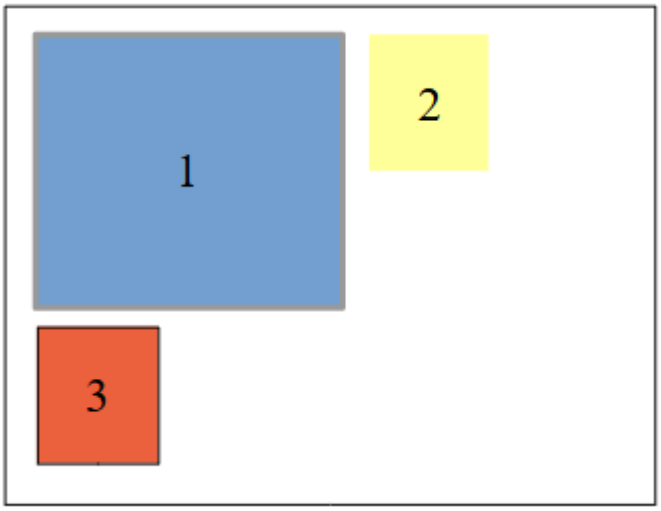


Ejercicio 2:

Crea un documento HTML llamado **cajas.html**. Dentro, crea tres cajas (tres div) y, con estilos CSS cargados de un documento aparte llamado **estilos.css**, haz que:

- La primera caja tenga color de fondo azul, borde gris continuo de 2 píxeles de grosor por todo el contorno y margen derecho e inferior de 10 píxeles. Debe tener una anchura (*width*) del 50% y una altura (*height*) de 100 píxeles.
- La segunda caja tenga color de fondo amarillo claro, sin borde. Debe tener una anchura del 20% y una altura de 50 píxeles.
- La tercera caja debe tener color de fondo rojo claro, con borde negro doble de 1 píxel de grosor. Debe tener una anchura del 20% y una altura de 50 píxeles.
- Las tres cajas deben tener un posicionamiento flotante a la izquierda, pero la tercera debe tener una propiedad `clear` para estar alineada con el margen izquierdo.

Al final, debe quedarte algo parecido a esto (aunque con otras proporciones seguramente):



Prueba a cambiar el tamaño de la ventana para comprobar cómo se ajusta la anchura de las cajas al nuevo tamaño.

3.1.1. Otros parámetros de posición

Podemos utilizar las propiedades `top`, `bottom`, `left` y/o `right` para indicar una distancia respecto a los cuatro bordes de la página. Por ejemplo, el siguiente *div* queda posicionado justo en la esquina inferior izquierda:

```
div
{
    position: fixed;
    bottom: 0px;
    left: 0px;
}
```

3.2. Posicionamiento con rejilla (*grid*)

El modo de posicionamiento anterior ha sido muy utilizado durante muchos años, pero tiene el inconveniente de su complejidad: hay que ajustar bien las dimensiones y el posicionamiento de las cajas para evitar que se nos descuadren en una web.

Como alternativa, desde hace unos pocos años, se ha implantado un sistema de posicionamiento CSS basado en una rejilla. Esto, junto con otras herramientas CSS que han surgido en los últimos años como Flexbox, permite colocar los componentes de la página de una forma más flexible y limpia.

Estructura del documento

Para empezar a trabajar con CSS Grid, debemos ubicar todos los elementos que queramos posicionar dentro de un elemento contenedor, que puede ser un *div*, o un *section*, por ejemplo. A este elemento contenedor lo identificamos con una clase (*class*) o con un *id*, si va a ser único en la página.

```
<section id="contenedor">
    <div id="elemento1">Uno</div>
    <div id="elemento2">Dos</div>
    <div id="elemento3">Tres</div>
    <div id="elemento4">Cuatro</div>
    ...
</section>
```

Configuración de la rejilla

Después, debemos activar *grid* en el elemento contenedor. Para ello, haremos uso de una propiedad llamada `display`, y le pondremos el valor de *grid*:

```
#contenedor
{
    display: grid;
}
```

Para establecer el tamaño o número de celdas de la rejilla podemos hacer uso de las siguientes propiedades CSS:

- `grid-template-columns`: indicamos, separados por espacios, los tamaños de cada columna de la rejilla. Podemos utilizar las unidades habituales (porcentajes, píxeles, etc) o utilizar una unidad propia de CSS Grid llamada `fr` (*fraction*), que se refiere a fracciones del espacio disponible. Así, una columna de *2fr* será el doble de ancha que una columna de *1fr*.
- `grid-template-rows`: indicamos, separados por espacios, los tamaños de cada fila de la rejilla. Aquí la unidad *fr* ya no se utiliza, porque el alto disponible no está delimitado, y solemos echar mano de alturas en píxeles o de alturas de ajuste automático al contenido (*auto*).

Por ejemplo, así establecemos una rejilla de 3 columnas y 2 filas en el contenedor. La columna central será el doble de ancha que el resto. La primera fila tiene una altura fija de 50 píxeles, y la segunda tiene un ajuste automático al contenido.

```
#contenedor
{
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: 50px auto;
 }
```

El espaciado entre celdas lo podemos definir con la propiedad `gap`:

```
#contenedor
{
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: 50px auto;
    gap: 10px;
 }
```

Notar que las anchuras y alturas también pueden darse en otras unidades, como indicábamos. Por ejemplo, las tres columnas del ejemplo anterior también podríamos definirlas así:

```
#contenedor
{
    ...
    grid-template-columns: 25% 50% 25%;
    ...
}
```

Ubicar elementos en la rejilla

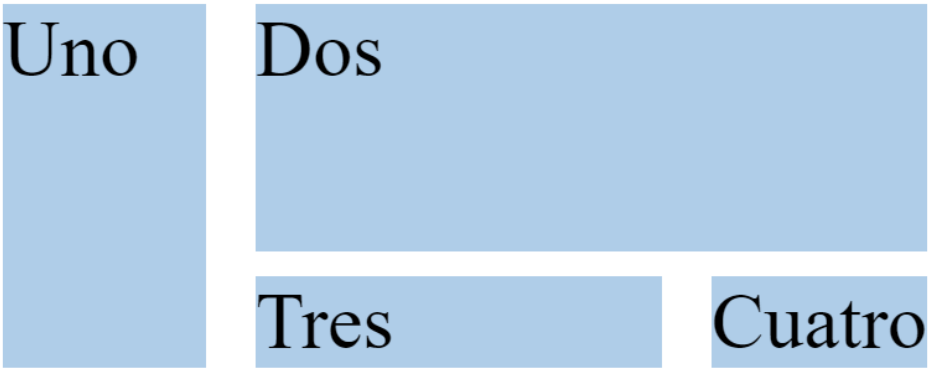
Si no indicamos en CSS la ubicación de cada elemento, éstos se colocarán por defecto de izquierda a derecha y de arriba a abajo, ocupando cada uno una celda de la rejilla. Alternativamente, podemos utilizar propiedades como `grid-row-start`, `grid-row-end`, `grid-column-start` y `grid-column-end` para ubicar cada elemento en la rejilla. Las filas y columnas de fin (*grid-row-end* y *grid-column-end*) indicadas quedarían sin ocupar. Por ejemplo, de este modo indicamos que el primer elemento ocupe la primera casilla de la rejilla en las dos primeras filas, y el segundo elemento ocupe las otras dos casillas de la primera fila:

```
#elemento1
{
    grid-row-start: 1;
    grid-row-end: 3;
    grid-column-start: 1;
    grid-column-end: 2;
}

#elemento2
{
    grid-row-start: 1;
    grid-row-end: 2;
    grid-column-start: 2;
    grid-column-end: 4;
}
```

Como alternativa, podemos utilizar la notación *X / Y* para indicar, con las propiedades `grid-row` y `grid-column` la posición del elemento. En este caso *X* hace referencia a la fila o columna de inicio (inclusive), e *Y* hace referencia a la fila o columna de fin (exclusive). Si sólo se ocupa una fila o una columna, el elemento *Y* puede omitirse. El siguiente fragmento CSS sería equivalente al anterior:

```
#elemento1
{
    grid-row: 1 / 3;
    grid-column: 1;
}

#elemento2
{
    grid-row: 1;
    grid-column: 2 / 4;
}
```

Así quedarían ubicados los elementos anteriores en la rejilla definida (los marcamos con fondo azul para identificarlos mejor). Notar como los elementos "Tres" y "Cuatro" simplemente ocupan una casilla cada uno, a continuación de los anteriores, ya que para ellos no hemos definido ninguna disposición.



> **Ejercicio 3:**
>
> Crea un documento HTML llamado **ofertas.html**, y trata de simular esta apariencia usando CSS Grid, bordes y otras propiedades CSS que hemos estado viendo hasta ahora:

> Algunas pistas:

> - La página tiene un margen lateral del 20% a cada lado, y fuente Arial en todo el documento
> - El color empleado para los bordes, el anuncio de prueba de versión profesional y los precios es azul (*blue*).

3.3. La propiedad *display*

Hemos utilizado en algún ejemplo anterior una propiedad CSS llamada `display`. Mediante esta propiedad podemos determinar cómo queremos que se disponga un elemento en una página. Algunos de los valores más habituales que podemos darle son:

- **block**: queremos que el elemento se comporte como un elemento en bloque. Este valor es útil cuando el elemento no es un elemento en bloque (por ejemplo, un enlace, o una imagen) y queremos que se comporte como tal
- **inline**: queremos que se comporte como un elemento en línea. Al contrario que el anterior, esta opción puede resultar útil para hacer que elementos en bloque (como párrafos, o items de listas) se comporten como elementos en línea. Veremos un ejemplo a continuación.
- **inline-block**: es una mezcla de las dos anteriores. Permite a un elemento en bloque comportarse como un elemento en línea, pero conservando algunas propiedades de elemento en bloque, como por ejemplo, y sobre todo, la posibilidad de definir su anchura y altura.
- **none**: oculta el elemento en la página
- **hidden**: similar al anterior, pero el elemento sigue conservando su espacio (no se ve, pero se mantiene el espacio que ocupa)
- **grid**: para definir rejillas CSS con sus diferentes filas y columnas, como hemos visto en un ejemplo anterior.

4. Estilos para listas

Ya hemos visto en documentos anteriores que existen diferentes tipos de listas HTML (ordenadas, no ordenadas y de definición) y que las etiquetas disponen de una serie de atributos para, por ejemplo, elegir el tipo de numeración (letras, números romanos, etc), el número de comienzo, tipo de viñeta, etc.

Mediante CSS podemos aplicar los estilos generales que hemos visto hasta ahora para texto y párrafo (tipo de letra, interlineado, etc), pero también existen una serie de propiedades específicas para el estilo de las listas en sí.

`list-stype-type`

Esta propiedad permite elegir el tipo de viñeta (listas no ordenadas) o numeración (listas ordenadas) que aplicar en las listas. Sería la alternativa al atributo `type` en las etiquetas `ol` o `ul`.

En el caso de listas no ordenadas, algunos de los valores posibles son *disc*, *circle*, *square*... Para las listas ordenadas, podemos elegir entre numeración normal (*decimal*), romanos en minúscula (*lower-roman*), en mayúscula (*upper-roman*), letras en minúscula (*lower-alpha*), mayúscula (*upper-alpha*), etc. Aquí podéis encontrar información más detallada sobre los valores de esta propiedad.

En este ejemplo, definimos que las listas ordenadas estarán definidas por números romanos en mayúscula:

```
ol
{
    list-style-type: upper-roman;
}
```

`list-style-position`

Esta propiedad define la posición de la viñeta (o numeración) respecto al contenido del ítem. Sus valores pueden ser *inside* (la viñeta o numeración se añade junto al contenido del ítem, lo que provoca que todo quede alineado a la izquierda), o *outside* (la viñeta o numeración queda fuera del ítem, con lo que las viñetas se alinean por su cuenta, y los items por la suya. Es el valor por defecto).

Por ejemplo, esta lista ordenada con números romanos queda definida con posición *inside*:

```
ol
{
    list-style-type: upper-roman;
    list-style-position: inside;
}

...

<ol>
    <li>Uno</li>
    <li>Este es el dos</li>
    <li>Este el tres</li>
    <li>Y, para finalizar, este es el cuatro</li>
</ol>
```

Obtendríamos este resultado:

I. Uno
II. Este es el dos
III. Este el tres
IV. Y, para finalizar, este es el cuatro

En cambio, si la definimos con posición *outside* (o no indicamos posición)...

```
ol
{
    list-style-type: upper-roman;
}
```

... obtendríamos este otro resultado:

I. Uno
II. Este es el dos
III. Este el tres
IV. Y, para finalizar, este es el cuatro

`list-style-image`

Esta propiedad permite definir nuestra propia imagen como viñeta, y utilizarla en listas no ordenadas.

```
ul
{
    list-style-image: url("imagenes/estrella.svg");
}
```

Sin embargo, la propiedad está algo limitada en algunos aspectos. Por ejemplo, no podemos controlar el tamaño de la imagen a mostrar (tendríamos que guardarla ya escalada al tamaño que queramos mostrar), y tampoco podríamos controlar su posición. Como alternativa, podemos definir un estilo de viñeta nulo (*none*) sobre la lista en general, y poner un estilo sobre cada item de la lista (*li*) indicando que queremos ponerle una imagen de fondo sin repetir, que hará de viñeta. En este caso, podemos especificar la posición y tamaño de la imagen. Por ejemplo:

```
ul
{
    list-style-type: none;
}

ul li
{
    padding-left: 10px;
    background-image: url("imagenes/estrella.svg");
    background-position: 0 0;
    background-size: 1.6rem 1.6rem;
    background-repeat: no-repeat;
}
```

4.1. Listas y posicionamiento

Es muy habitual utilizar listas para definir los elementos de un menú de navegación (dentro de una etiqueta `nav`, por ejemplo). En este caso, los elementos suelen estar adyacentes, uno junto a otro. Pero, como hemos visto, las listas son elementos de bloque, en las que, por defecto, los ítems se colocan uno debajo del otro. Sin embargo, podemos utilizar la propiedad `display` de los items de la lista para indicar que queremos que se comporten como elementos en línea. Además, podemos definir la propiedad `list-style-type` como `none` para que no tengan ningún tipo de viñeta. Uniendo ambas cosas, podemos definir un menú de navegación horizontal. Aquí vemos un ejemplo:

```
nav
{
    background-color: #CCC;
}

nav ul.navegacion
{
    list-style-type: none;
    text-align: center;
}

nav ul.navegacion li
{
    display: inline;
    margin: 20px;
}
```

Y lo podríamos aplicar de este modo a una lista en una barra de navegación:

```
<nav>
    <ul class="navegacion">
        <li>Inicio</li>
        <li>Preguntas frecuentes</li>
        <li>Tienda online</li>
        <li>Contacto</li>
    </ul>
<nav>
```

El resultado sería algo así:

Inicio Preguntas frecuentes Tienda online Contacto

> **Ejercicio 4**

En este enlace puedes descargar el contenido HTML de una página web que simula la página de inicio de un instituto. Hay un título principal de la página, un menú de navegación, un apartado de oferta formativa, y otro de novedades. Finalmente, hay una sección de pie de página con información de contacto del instituto. Se propone añadir el CSS necesario (desde un archivo aparte) para que se vea de este modo:

Algunas pistas:

- El color de fondo del encabezado y el pie es `#84250C` .
- El encabezado está ubicado pegado a la esquina superior izquierda, y el pie a la esquina inferior izquierda.
- La zona entre encabezado y pie tiene un margen superior de 150px, y lateral de 20% a cada lado.