Materiales del curso N

1.5 Posicionamiento

Contenidos

Introducción

En esta sesión veremos dos recursos para modificar la apariencia de nuestra web: visualización (o display, que es fundamental) y posicionamiento. Ambos permiten modificar cómo se muestran los elementos de la página ya sea modificando su tamaño, su posición o ambos a la vez.

¿Para qué sirve lo que vamos a ver en esta sesión?

Para saber las diferentes opciones que tenemos de colocar los elementos según el diseño que nos manden.

¿En qué casos se utiliza?

Cuando tengamos que darle un aspecto determinado al contenido, es decir, siempre :)
Algunos casos concretos son:

- Un listado de elementos distribuidos por columnas.
- El típico módulo que lleva el corazón de "like" en una esquina.
- Una galería donde las flechas de anterior/siguiente estén, una cada lado, y centradas verticalmente.
- El típico módulo de precios "desde 5€ al mes" donde el diseñador ha desplegado toda su creatividad compositiva con un diseño con elementos a diferentes tamaños.
- El menú que se mantiene en la parte superior del navegador al hacer *scroll*.
- El menú que aparece por uno de los laterales.

Objetivos de la sesión

1. Recordar los principales modos de presentación de los elementos HTML: `inline`, `inline-block` y `block`.
2. Conocer los modos de posicionamiento: `static`, `relative`, `absolute` y `fixed`.

Visualización (display)

Antes de meternos con el posicionamiento recordemos los modos de presentación que vimos en la sesión 1.3:

La propiedad CSS `display`, se encarga de definir cómo se va a visualizar un elemento HTML, cómo va a colocarse en la página y cómo se colocarán el resto de elementos respecto a este. Según el valor que tenga asignado display, un elemento puede ocupar el ancho entero de su contenedor, ocupar solo el espacio que necesite para mostrar su contenido, mostrarse como si fuese una casilla de una tabla o directamente ocultarse.

Los navegadores web aplican por defecto un valor `display` a todos los elementos HTML de nuestra web. Hay muchos valores distintos para `display` pero, por el momento, nosotros solo veremos cuatro:

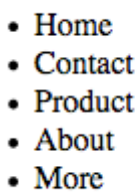
- **Block:** Los elementos en bloque se muestran ocupando el ancho completo de su contenedor. Los elementos en bloque siempre empiezan en una nueva línea y nunca van a tener más elementos a su misma altura dentro del mismo contenedor, estarán más arriba o más abajo. Se les puede aplicar margin, padding, alto y ancho.
- **Inline:** Los elementos en línea o inline son aquellos que ocupan lo que ocupa su contenido. En estos, el tamaño será exactamente el tamaño de su contenido y no podremos asignarle un tamaño diferente, ni tendrá efecto un margin vertical.
- **Inline-block:** Los elementos inline-block ocupan por defecto el ancho de su contenido y se comportan como si se tratase de un elemento en línea, pero permiten tener un ancho, un alto y relleno y márgenes verticales, como sucede con los elementos en bloque.
- **None**: Oculta por completo cualquier elemento al que se lo apliquemos, será como si ese elemento no existiese ya que no se mostrará y el resto de elementos de la página lo ignorarán.

EJERCICIO 1

Listas horizontales

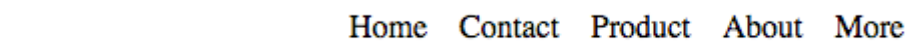
Crea una lista de cinco elementos que se muestre en línea y con espacios entre cada elemento de 12 píxeles.

Vista de la lista sin estilos:



Menu sin estilos

Vista de la lista con estilos:



Menu con estilos

Posicionamiento de elementos

Antes de tener disponible flexbox el posicionamiento que vamos a ver era la principal opción de solucionar diseños complejos. En cualquier caso todavía puede que nos encontremos con soluciones muy complejas o código ya resuelto con este sistema, así que vamos a verlo:

Aparte de modificar la distribución, podremos hacer que cambien su comportamiento a la hora de hacer scroll en la página y que modifiquen la posición de otros objetos al modificar la suya propia.

La propiedad `position` de CSS será la que nos permita modificar la forma en la que se distribuyen los objetos a través de una página web.

El atributo `position` es fundamental en casos muy concretos con las webs actuales, es decir, no podríamos tener ciertas superposiciones y no podríamos sacar a un elemento del flujo de la página para que el resto de elementos (contenedor y elementos hermanos) no lo tengan en cuenta y lo ignoren.

La propiedad `position` tiene cuatro valores posibles:

- `static` : Es el tipo de posición por defecto en todos los elementos HTML.
- `relative` : Permite modificar la posición de un elemento en función de su posición actual en la página.
- `absolute` : Saca al elemento del flujo de la página, es decir, hace que su contenedor y los elementos de antes y después no lo tengan en cuenta a la hora de posicionarse y definir su tamaño y por otro lado posiciona el elemento en función de la posición del body o en su defecto del primer elemento contenedor que tenga una posición diferente a static (posición por defecto).

- `fixed` : Saca a un elemento del flujo normal de la página y permite posicionarlo en función de la ventana del navegador. Aparte, este tipo de elementos mantienen su posición cuando hacemos *scroll* en la página (como si se mantuviesen anclados en un mismo punto), de ahí su nombre fixed (fijo).

Algunos recursos en video:

- Video para entender el posicionamiento web
- Video para entender position: static
- Video para entender position: relative
- Video para entender position: absolute

Como hemos visto, cuando posicionamos una caja con cualquier valor que no sea static y modificamos su posición horizontal y/o vertical (top, right, bottom, left) esta se puede superponer visualmente por encima de otras.

Esto ocurre porque, adicionalmente a sus posiciones horizontales y verticales, las cajas se apilan a lo largo de un "eje z".

Cuando las cajas se superponen por encima de otras, se están posicionando en capas adicionales a la capa normal de renderizado (capa 0).

La posición Z de cada capa representa el orden de apilamiento. Podemos modificarlo con la propiedad **z-index**. Números más grandes significan mayor cercanía a la observadora.

- Video para entender z-index.

EJERCICIO 2

Desplazando divs relativamente

Define un documento HTML con un div padre (divPadre), dentro del cual existan otras 3 cajas contenedoras div (div1, div2 y div3), cada una de ellas con unas dimensiones de 300x300px, 40 píxeles de margin en todas direcciones, 30 píxeles de padding en todas direcciones y un background color diferente. Usando posicionamiento relativo genera un desplazamiento de los div de la siguiente manera:

1. El div 1 deberá desplazarse 100 píxeles a la derecha y 50 píxeles hacia abajo respecto a lo que sería su posición normal.
2. El div 2 deberá desplazarse 150 píxeles a la izquierda y 320 píxeles hacia arriba respecto a lo que sería su posición normal.
3. El div 3 deberá desplazarse 180 píxeles a la derecha y 240 píxeles hacia arriba respecto a lo que sería su posición normal.

EJERCICIO 3

Vente conmigo

Crear un documento HTML con una cabecera y un contenedor principal con varios párrafos que contengan suficiente texto como para que la página se muestre con scroll (barras de desplazamiento).

1. Añadir fondo morado a la cabecera y hacer que se mantenga fija arriba.
2. Hacer que la cabecera no tape el contenedor principal cuando no hemos hecho scroll, sin utilizar `margin` ni `padding` (PISTA: posiciona el contenedor principal).
3. Hacer que cuando la usuaria haga scroll, la cabecera se apile o superponga por encima del contenedor principal.

Transform

Otra forma interesante de modificar la posición de un elemento HTML es la propiedad `transform` con la que podemos realizar una serie de ajustes al elemento respecto a sí mismo. Sin embargo, no saca al elemento del flujo de la página como `position: absolute` o `position: fixed`, y el resto de elementos de la página se comportan como si no hubiésemos aplicado una transformación a uno de ellos.

Vamos a ver varias transformaciones:

- Translate
- Scale
- Rotate

Translate

Permite desplazar el elemento, una cantidad dada, horizontal y/o verticalmente:

```
.element {
  transform: translate(-100px, 100px);
}
```

Scale

Permite agrandar o reducir el elemento proporcionalmente, o indicando las deformaciones horizontal y vertical independientemente:

```
.element {
  transform: scale(2);
}
```

Rotate

Permite rotar el elemento:

```
.element {
  transform: rotate(45deg);
}
```

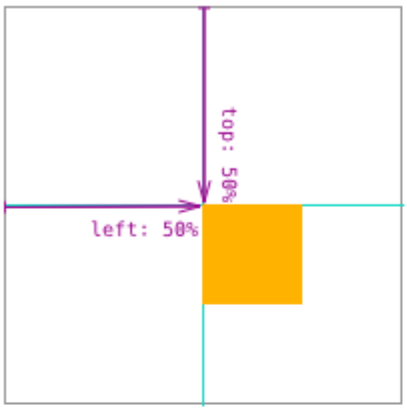
EJERCICIO 4

Transformers

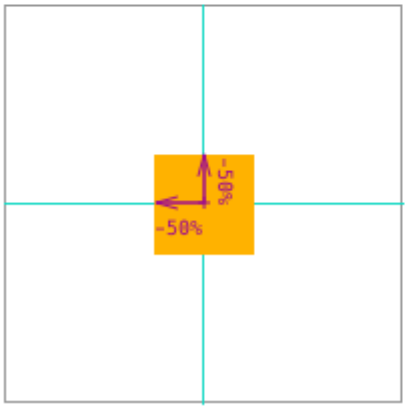
¿Sabrías resolver estos ejercicios de transformaciones tras leerte la documentación de transform?

▸ Los ejercicios de transform en codepen

`Transform` se suele usar mucho junto con posiciones fijas o absolutas para centrar elementos horizontal y verticalmente:




```
.orange {
  left: 50%;
  position: asbolute;
  top: 50%;
}
```




```
.orange {
  left: 50%;
  position: asbolute;
  top: 50%;
  transform: translate(-50%, -50%);
}
```

Ejemplo de centrado con position y transform

EJERCICIO 5

Cajas sobre cajas

Vamos a definir un documento HTML con 3 cajas contenedoras `div` (div1, div2 y div3):

- La primera con unas dimensiones de 500x500px y color de fondo amarillo.
- La segunda con dimensiones 300x300px y color de fondo verde.

- La tercera con dimensiones 150x150px y color de fondo azul.

Usando posicionamiento absoluto establecemos para el div2 y el div3 el mismo origen que para el div1, de modo que las cajas se superpongan y el efecto generado sea ver cuadrado azul sobre uno verde que a su vez está sobre uno amarillo.

Ahora tenemos que hacer que las cajas estén centradas vertical y horizontalmente añadiendo:

- 40px de padding y 2px de borde al div1.
- 75px de padding al div2.
- 20px de borde de puntos al div3.

Para esto usad `box-sizing: border-box;` .

EJERCICIO 6

Aviso de cookies

Vamos a definir un documento HTML con varios `div` s que contengan suficiente texto como para que la página se muestre con scroll (barras de desplazamiento). El último de los `div` s debe:

- contener el texto *"Esta página web utiliza cookies. Si continúa navegando acepta el uso de cookies."*.
- un valor `height` (altura) de 100 píxeles y color de fondo amarillo.

Usando posicionamiento fixed, tenemos que fijar este `div` en la parte inferior de la página de modo que se visualize siempre, aún cuando hagamos scroll.

EJERCICIO 7

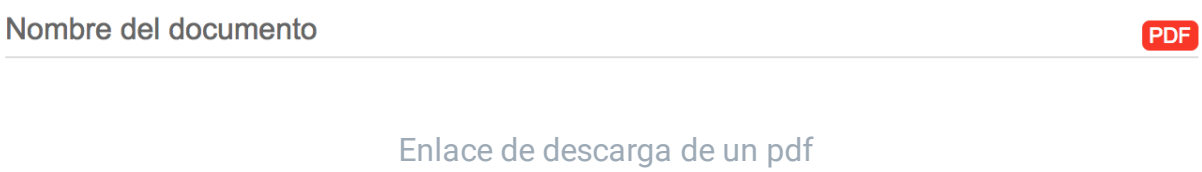
Ese texto necesita aire

Crea un texto que ocupe el 86% de la pantalla y esté centrado dentro del body. Usaremos la propiedad `max-width` para dar un ancho máximo de 600px. Más info acerca de max-width.

EJERCICIO 8

Dame PDFs

Hacer un enlace de descarga de un archivo (por ejemplo PDF) con una etiqueta que refleje el tipo del archivo y que siempre esté a la derecha.



Enlace de descarga de un pdf

EJERCICIO 9

Esto es un artículo

Crea una composición similar a la de la imagen.

Muestra

Las dimensiones de esta composición serían las siguientes:

- El body tendrá un borde de 8px
- El contenido estará centrado dentro del body e irá dentro de un `div` que tendrá 106px de margen superior
- En el primer texto irá el nombre del autor con una fuente de 18px y un margen inferior de 40px
- El titular irá después con un tamaño de fuente de 32px y un margen inferior de 32px
- Cada párrafo tendrá un tamaño de fuente de 18px y un margen inferior de 27px
- El enlace tendrá un padding superior e inferior de 8px y otro izquierdo y derecho de 16px y un margen izquierdo de -16px

---

EJERCICIO 10

Tecnologías web

La web que vamos a crear consta de las siguientes características:

1. Toda la web usa una tipografía sin serifa (`sans-serif`)
2. Tiene como título "Tecnologías web"
3. Tiene un párrafo que describe qué son las tecnologías web
4. Al final del párrafo, tiene un listado de tecnologías compuesto por: HTML, CSS y JavaScript, cada una de las cuales aparece subrayada para indicar que se puede interactuar
5. Al poner el ratón sobre cualquiera de ellas:
    1. el cursor cambia para indicar que estamos obteniendo ayuda
    2. aparece un tooltip (recuadro flotante de 400px por 200px) de color blanco, con el nombre de la tecnología como título del tooltip y una breve descripción de la misma

---

EJERCICIO 11

Una web sencilla

Vamos a crear una web sencilla, con las siguientes características:

1. Una cabecera que consta únicamente de un título
2. Un cuerpo principal que consta de un montón de párrafos, tantos como para que la página tenga scroll
3. Un pie de página con
    1. El nombre de la empresa
    2. Un listado de redes sociales de la empresa (en formato texto o imagen) que aparezcan en línea
4. Un texto para indicar que el sitio web usa cookies con un enlace para ver más info, que aparece en la esquina inferior derecha de la pantalla y que sigue ahí al hacer scroll

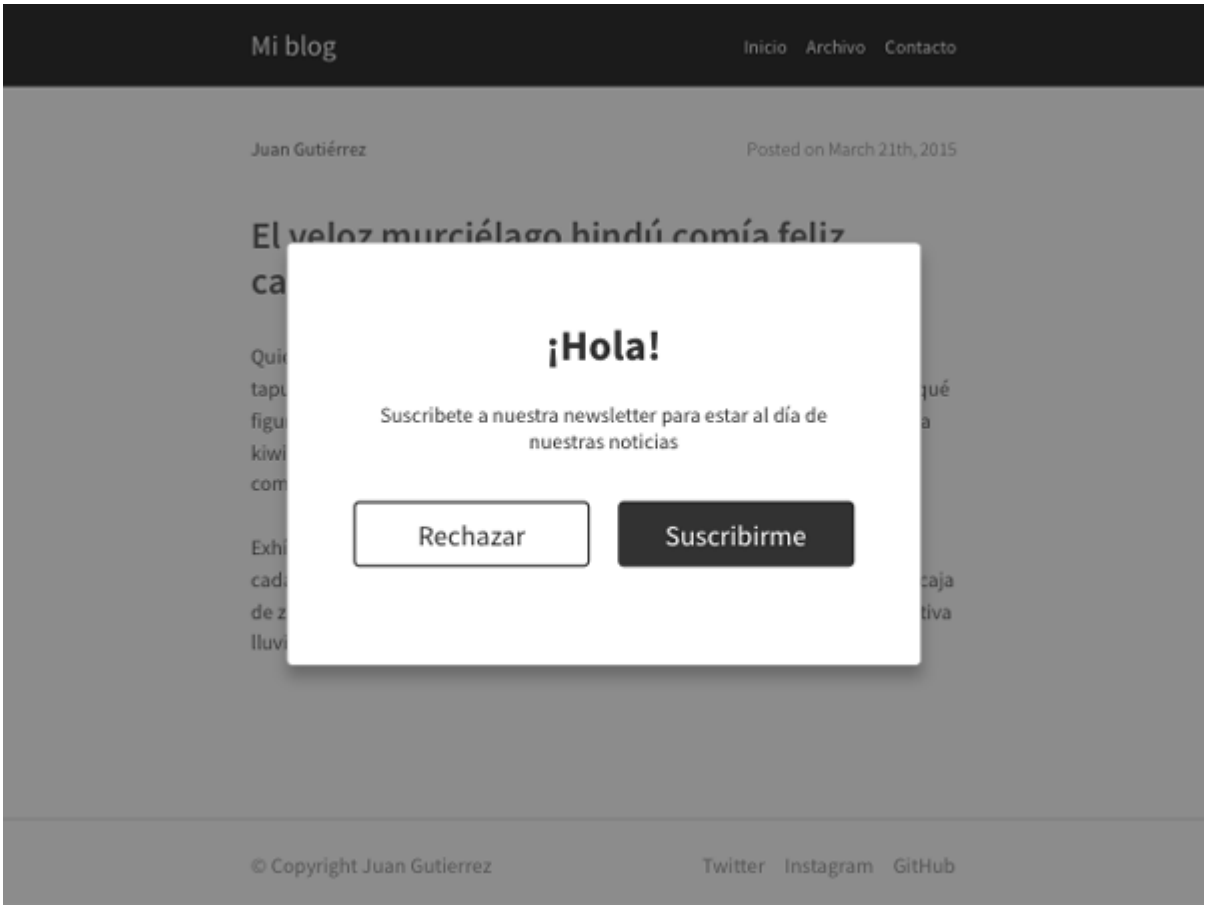
---

EJERCICIO 12

Con buenos modales

Vamos a continuar desde el ejercicio 8 de la lección de flexbox y vamos a añadir una modal.

Crearemos un elemento que se superponga sobre él. Ese elemento tendrá un fondo oscuro transparente y un `div` en su interior. Ese `div` entero estará centrado tanto vertical como horizontalmente y contendrá un titular, un texto y un par de botones. El resultado debe ser similar a la siguiente imagen.



Ejemplo

En la imagen, es importante observar que la cabecera estará por debajo de la ventana emergente.

Bonus

Devdocs

DevDocs va a ser nuestra página de documentación de cabecera ya que agrupa documentación oficial de diferentes temas de front: lenguajes, tecnologías, preprocesadores...

Os proponemos DevDocs porque es un agrupador de documentación oficial que te permite tener contenido para consultar sin conexión en tu navegador.

Recursos externos

- Cómo funciona float
- Libro Introducción a CSS - 5.1. Tipos de elementos
- Libro Introducción a CSS - 5.2. Posicionamiento
- Libro Introducción a CSS - 5.3. Posicionamiento normal (static)
- Libro de Introducción a CSS - 5.4 Posicionamiento relativo
- Libro de Introducción a CSS - 5.5 Posicionamiento absoluto
- Libro de Introducción a CSS - 5.6 Posicionamiento fijo
- Libro de Introducción a CSS - 5.7 Posicionamiento flotante