

## CONTENTS

Primera parte.....	2
Pregunta 1. Sobre posicionamiento, responde a las siguientes cuestiones: .....	2
1. ¿Para qué sirve la propiedad <code>clear</code> ? Indica brevemente el significado de sus diferentes valores, poniendo un ejemplo de uso de cada uno de ellos. ....	2
2. Escribe el código CSS necesario para conseguir que una imagen de nombre <code>logo.png</code> aparezca en la esquina superior derecha de una página, permaneciendo siempre visible, aunque se haga scroll vertical a dicha página.....	4
Pregunta 2. Sobre el modelo de caja, responde a las siguientes cuestiones: .....	4
1. ¿Cómo afecta la propiedad <code>box-sizing</code> al modelo tradicional de caja de CSS? ¿Cuáles son sus principales valores? Pon un ejemplo de uso de cada uno de ellos que ilustre sus diferencias.....	4
2. En el archivo <code>ColumnasDiferenteAltura.html</code> planteamos el siguiente problema: tenemos dos columnas posicionadas con "float" que tienen diferente altura ya que adquieren la altura que fuerza su propio contenido. Debes encontrar dos soluciones que permitan que las dos columnas tengan la misma altura (en la imagen <code>ColumnasIgualAltura.png</code> , puedes ver el resultado que debes conseguir). Las soluciones que aportes en ningún caso deben contemplar la posibilidad de utilizar el recurso de dar a ambas la misma altura con <code>height</code> . Debes presentar tus soluciones en dos archivos diferentes: <code>ColumnasSolucion1.html</code> y <code>ColumnasSolucion2.html</code> , el CSS incrustado en el head de cada uno de ellos.....	6
Pregunta 3. Sobre diseño web adaptativo, responde a las siguientes cuestiones: .....	7
1. ¿En qué consiste la filosofía <code>mobile first</code> ? ¿Qué cambios crees que implica a la hora de abordar el diseño de una página web? .....	7
2. ¿Cómo podemos conseguir que una imagen sea <code>responsive</code> ? Investiga sobre ello y haz un pequeño resumen de las principales técnicas que lo permiten, indicando sus ventajas e inconvenientes.....	7

## PRIMERA PARTE

## PREGUNTA 1. SOBRE POSICIONAMIENTO, RESPONDE A LAS SIGUIENTES CUESTIONES:

1. ¿PARA QUÉ SIRVE LA PROPIEDAD `clear`? INDICA BREVEMENTE EL SIGNIFICADO DE SUS DIFERENTES VALORES, PONIENDO UN EJEMPLO DE USO DE CADA UNO DE ELLOS.

La propiedad `clear` indica el lado o lados de una caja que no debe ser adyacente a un elemento posicionado de forma flotante con la propiedad `float`. Los cuatro valores permitidos para esta propiedad tienen el siguiente significado:

- **left**, hace que la caja sobre la que se aplica baje hasta que su borde superior esté por debajo del borde inferior de cualquier elemento flotado a la izquierda.
- **right**, hace que la caja sobre la que se aplica baje hasta que su borde superior esté por debajo del borde inferior de cualquier elemento flotado a la derecha.
- **both**, hace que la caja sobre la que se aplica baje hasta que su borde superior esté por debajo del borde inferior de cualquier elemento flotado a la izquierda o a la derecha.
- **none**, es el valor por defecto que se aplica a todos los elementos y no tiene efecto sobre la posición de la caja.

Para entender mejor el concepto, veamos el siguiente ejemplo:

```
<body>
  <div class="caja1"> Este es el contenedor 1</div>
  <div class="caja1"> Este es el contenedor 2</div>
  <div class="caja2"> Este es el contenedor 3</div>
  <div class="caja1"> Este es el contenedor 4</div>
</body>
```

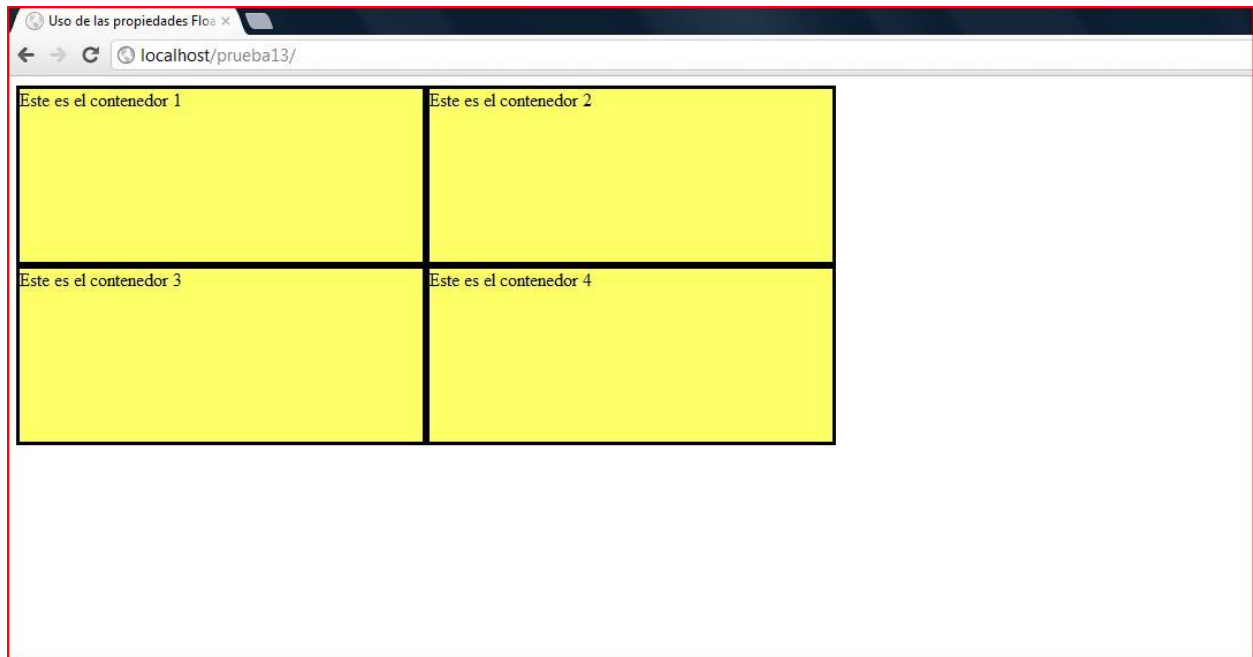
Se puede observar que hemos definido una clase llamada `caja2`, en la cual utilizamos la propiedad `clear` y le asignamos el valor de `left`. Después de utilizar la propiedad `clear:left`, asignamos la propiedad `float:left` para esa clase, es decir, los elementos que la utilicen comenzarán un nuevo esquema de posicionamiento pero ya en una línea inferior.

A continuación se presenta un ejemplo con los 4 contenedores, los 2 primeros utilizan la clase `caja1`, la cual tiene activada la propiedad `float:left`, por lo que el contenedor 1 se colocará hasta la izquierda de la página y enseguida de él se colocará el contenedor 2, quién intenta hacer lo mismo.

Pero podemos observar que el tercer contenedor utiliza la clase `caja2`, la cuál utiliza la propiedad `clear:left`, al hacer esto le estamos indicando que rompa el esquema de posicionamiento de `float:left` que se venía utilizando y que comience uno nuevo, en una nueva línea. El contenedor 4 al utilizar la clase `caja1`, intentará colocarse hasta la izquierda, pero del nuevo esquema creado, es decir, a un lado del contenedor 3. Esto queda de la siguiente manera

```
.caja1{
  width: 350px;
  height: 150px;
  background: #FF0000;
  border:solid #000;
  float: left;
}

.caja2{
  width: 350px;
  height: 150px;
  background: #FF0000;
  border:solid #000;
  float: left;
  clear: left;
}
```



2. ESCRIBE EL CÓDIGO CSS NECESARIO PARA CONSEGUIR QUE UNA IMAGEN DE NOMBRE `LOGO.PNG` APAREZCA EN LA ESQUINA SUPERIOR DERECHA DE UNA PÁGINA, PERMANECIENDO SIEMPRE VISIBLE, AUNQUE SE HAGA SCROLL VERTICAL A DICHA PÁGINA.

Archivo HTML

```
<!DOCTYPE html>
<html>

<head>
  <title></title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>

<body>
  
</body>

</html>
```

Archivo CSS

```
#logo {
  position: fixed;
  top: 0;
  right: 0;
}
```

## PREGUNTA 2. SOBRE EL MODELO DE CAJA, RESPONDE A LAS SIGUIENTES CUESTIONES:

1. ¿CÓMO AFECTA LA PROPIEDAD `BOX-SIZING` AL MODELO TRADICIONAL DE CAJA DE CSS? ¿CUÁLES SON SUS PRINCIPALES VALORES? PON UN EJEMPLO DE USO DE CADA UNO DE ELLOS QUE ILUSTRE SUS DIFERENCIAS.

El ancho de un elemento se altera si se le aplica un borde o un padding. Eso es porque la anchura del elemento que tu especificas con CSS, por defecto no incluye borde ni padding.

Un ejemplo: Éste es el efecto que tiene un padding y un borde sobre un elemento de 200px de ancho:

```
<div style="width:200px;
padding: 20px;
border: 10px solid #ccc;
margin: 0 auto;">
  Lorem ipsum...
</div>
```

Lorem ipsum dolor sit amet,  
consectetur adipisicing elit, sed  
do eiusmod tempor incididunt  
ut labore et dolore magna  
aliqua. Ut enim ad minim  
veniam, quis nostrud  
exercitation ullamco laboris...

Como se puede comprobar, no mide 200px de ancho, sino 260px. Es decir: 200px de ancho inicial, más 20px de padding izquierdo, más 20px de padding derecho, más 10px de borde izquierdo, más 10px de borde derecho.

Éste es el modo en el que los navegadores tratan los anchos por defecto. Sería el equivalente a

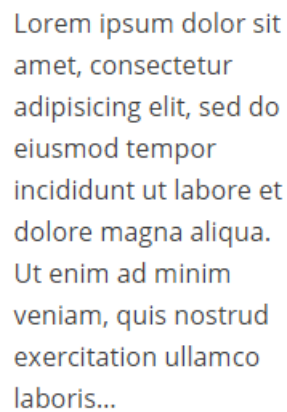
**box-sizing: content-box;**

---

#### BOX-SIZING: BORDER-BOX

Con border-box, hacemos que el ancho especificado sea el equivalente al ancho total. Es decir:

```
<div style="width: 200px;
padding: 20px;
border: 10px solid #ccc;
box-sizing: border-box;
margin: 0 auto;">
  Lorem ipsum...
</div>
```



Lorem ipsum dolor sit  
amet, consectetur  
adipiscing elit, sed do  
eiusmod tempor  
incididunt ut labore et  
dolore magna aliqua.  
Ut enim ad minim  
veniam, quis nostrud  
exercitation ullamco  
laboris...

Ahora ese elemento exactamente 200px, ni uno más, ni uno menos.

Esto es muy útil para elementos fluídos, cuando necesitas que el elemento ocupe (por ejemplo) un 33% del ancho, y si ocupa un píxel más toda la estructura se estropearía.

2. EN EL ARCHIVO `COLUMNASDIFERENTEALTURA.HTML` PLANTEAMOS EL SIGUIENTE PROBLEMA: TENEMOS DOS COLUMNAS POSICIONADAS CON "FLOAT" QUE TIENEN DIFERENTE ALTURA YA QUE ADQUIEREN LA ALTURA QUE FUERZA SU PROPIO CONTENIDO. DEBES ENCONTRAR DOS SOLUCIONES QUE PERMITAN QUE LAS DOS COLUMNAS TENGAN LA MISMA ALTURA (EN LA IMAGEN `COLUMNASIGUALALTURA.PNG`, PUEDES VER EL RESULTADO QUE DEBES CONSEGUIR). LAS SOLUCIONES QUE APORTES EN NINGÚN CASO DEBEN CONTEMPLAR LA POSIBILIDAD DE UTILIZAR EL RECURSO DE DAR A AMBAS LA MISMA ALTURA CON `HEIGHT`. DEBES PRESENTAR TUS SOLUCIONES EN DOS ARCHIVOS DIFERENTES: `COLUMNASSOLUCION1.HTML` Y `COLUMNASSOLUCION2.HTML`, EL CSS INCRUSTADO EN EL HEAD DE CADA UNO DE ELLOS.

Una posible solución es añadir la propiedad "display: flex" al container. Haciendo que todo lo que este dentro tenga la misma altura.

```
.container {  
  background-color: #000;  
  width: 90%;  
  max-width: 974px;  
  padding: 3%;  
  margin: 0 auto;  
  overflow: auto;  
  display: flex;  
}
```

La otra posible solución es, añadir un "bottom padding" muy grande y luego "engañar" al navegador añadiendo un margen negativo del mismo tamaño. Además hacer que el contenedor tenga "overflow: hidden"

```
.container {  
  background-color: #000;  
  width: 90%;  
  max-width: 974px;  
  padding: 3%;  
  margin: 0 auto;  
  overflow: hidden;  
}  
  
.columna {  
  float: left;  
  width: 46%;  
  margin-bottom: 3em;  
  padding: 2%;  
  padding-bottom: 500em;  
  margin-bottom: -500em;  
}
```

PREGUNTA 3. SOBRE DISEÑO WEB ADAPTATIVO, RESPONDE A LAS SIGUIENTES CUESTIONES:

1. ¿EN QUÉ CONSISTE LA FILOSOFÍA MOBILE FIRST? ¿QUÉ CAMBIOS CREES QUE IMPLICA A LA HORA DE ABORDAR EL DISEÑO DE UNA PÁGINA WEB?

Mobile First es una filosofía, una manera de encarar el trabajo y una forma de facilitar la labor durante el diseño responsive, comenzando siempre por los dispositivos, con pantallas menores. Mobile First" es un concepto bastante simple: diseñar pensando en los móviles primero.

El principal cambio es que se cambia el target para el que se hace la página web, es decir, hasta ahora las webs estaban diseñadas para pantallas de ordenador, pero ahora una gran parte de las páginas son visitadas a través de teléfonos móviles. Un cambio importante que está ocurriendo es en el esquema típico de página web, donde teníamos una cabecera, contenido, una o dos barras laterales y un pie de página. Ahora, si pensamos en "mobile first" las barras laterales deberían dejar de existir, no es cómodo. Otro aspecto importante es que tener muchas páginas en una misma web dificulta la navegabilidad, por las webs con una única página están en auge (también gracias a la aparición de frameworks como Angularjs).

2. ¿CÓMO PODEMOS CONSEGUIR QUE UNA IMAGEN SEA RESPONSIVE? INVESTIGA SOBRE ELLO Y HAZ UN PEQUEÑO RESUMEN DE LAS PRINCIPALES TÉCNICAS QUE LO PERMITEN, INDICANDO SUS VENTAJAS E INCONVENIENTES.

Una posible opción es utilizar las propiedades " width: 100%; height: auto;" y de esta forma hacer que la anchura de la imagen siempre ocupe el 100% de su contenedor.

Otra opción es utilizar el "srcset", que sirve para utilizar diferentes imágenes dependiendo de la resolución de la pantalla.

```
<img srcset="
  examples/images/image-384.jpg 1x,
  examples/images/image-768.jpg 2x
" alt="...">
```

La ventaja que tiene la primera opción es que la imagen se ajusta a la anchura de la pantalla, sin cambiar la imagen. Sin embargo la segunda opción permite que a cierta resolución se utilice una imagen distinta que sea más adecuada.

## DOCUMENTACIÓN

Para la realización de esta práctica he utilizado Bootstrap. Ya que creo que es muy útil a la hora de hacer que una web sea responsive y facilita mucho la maquetación. Bootstrap trae consigo archivos de estilo, por lo que he tenido que utilizar dos archivos, el mío “custom.css” y el del framework.

A la hora de organizar los html, he utilizado el esquema estándar. Es decir, en <head> se incluyen los meta, título, links a estilos y scripts. Luego, en el <body>, tengo un bloque “main” que engloba todo el contenido, dentro de él hay tres bloques grandes: cabecera, bodyCopy y footer.

La cabecera y el footer son comunes en todos los html. Para hacer que la cabecera sea responsive he utilizado la clase “collapse” y “collapsed” de Bootstrap. De esta forma, cuando la resolución es pequeña la barra de menú se convierte en un botón que al pulsar muestra las opciones.

Para hacer el footer he utilizado distintas clases para aplicar estilos. Para los iconos de las redes sociales he utilizado un script de la página fontawesome.com. El footer está dividido en dos bloques izquierda y derecha, en el de la izquierda se muestra la información de la empresa y en el de la derecha el formulario de suscripción.

Para el desarrollo del cuerpo de cada html he utilizado el grid de Bootstrap, se basa en que la página se divide en 12 columnas que puedes utilizar para organizar la información. Si quieres que algo ocupe el ancho total se utiliza 12 columnas, si quieres que sea la mitad entonces 6 columnas etc. Esto es muy útil a la hora de hacer responsive, ya que te permite, utilizando las clases correspondientes, que un bloque utilice las 12 columnas cuando la resolución es pequeña y 3-4 (o las que sean oportunas) cuando sea grande.

Para la página de útiles quería que fuera atractiva y una lista de links no lo es, por lo que me decante en hacer una especie de portfolio de los blogs y apps. Para ello cogí los 10 primeros blogs y cree “thumbnails” de ellos, cogiendo una foto de la web y un trozo de su descripción. Luego, para mostrar los demás blogs utilicé de nuevo el “collapse”. Para las apps, cree un “jumbotron” que al clicar redirige al link donde aparecen las apps y cree otros dos thumbnail para las dos primeras apps, con su logo y nombre.