Contents

[Primera parte 2](#_Toc471915014)

[Pregunta 1. Sobre posicionamiento, responde a las siguientes cuestiones: 2](#_Toc471915015)

[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. 2](#_Toc471915016)

[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. 4](#_Toc471915017)

[Pregunta 2. Sobre el modelo de caja, responde a las siguientes cuestiones: 4](#_Toc471915018)

[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. 4](#_Toc471915019)

[2. En el archivo ColumnasDiferenteAltura.html planteamos el siguiente problema: tenemos dos columnas posicionadas con "float". 6](#_Toc471915020)

[Pregunta 3. Sobre diseño web adaptativo, responde a las siguientes cuestiones: 7](#_Toc471915021)

[1. ¿En qué consiste la filosofía mobile first? ¿Qué cambios crees que implica a la hora de abordar el diseño de una página web? 7](#_Toc471915022)

[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. 7](#_Toc471915023)

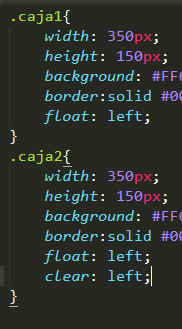
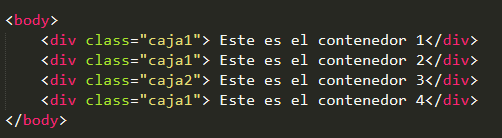
# Primera parte

## Pregunta 1. Sobre posicionamiento, responde a las siguientes cuestiones:

### ¿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:

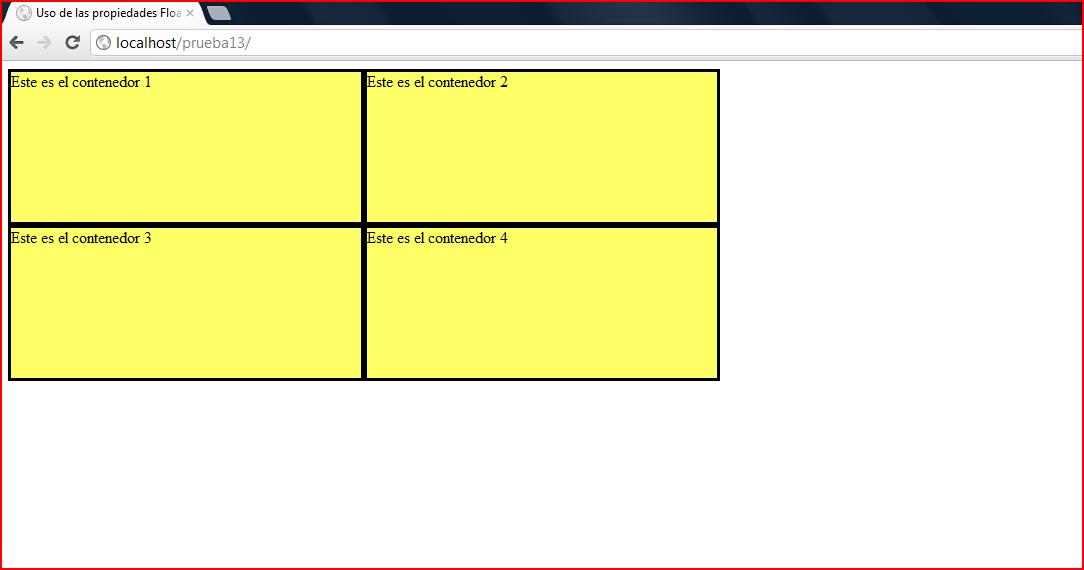
* **left**, hace que la caja sobre la que se aplica baje hasta que su borde superior esté por debajo del borde inferior de cualquier elemento flotado a la izquierda.
* **right**, hace que la caja sobre la que se aplica baje hasta que su borde superior esté por debajo del borde inferior de cualquier elemento 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:

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.

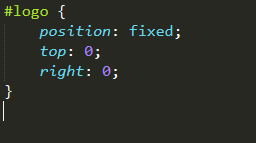
A continuación se presenta un ejemplo con los 4 cotenedores, los 2 primeros utilizan la clase caja 1, la cual tiene activada la propiedad float:left, por lo que el contenedor 1 se colocará hasta la izquierda de la página y enseguida de él se colocará el contenedor 2, quién intenta hacer lo mismo.

Pero podemos observar que el tercer contenedor utiliza la clase caja 2, 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 caja 1, intentará colocarse hasta la izquierda, pero del nuevo esquema creado, es decir, a un lado del contenedor 3. Esto queda de la siguiente manera



### 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 CSS

Archivo HTML

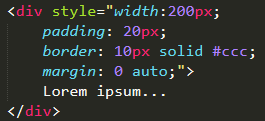
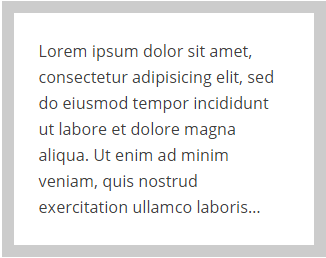


## Pregunta 2. Sobre el modelo de caja, responde a las siguientes cuestiones:

### ¿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:

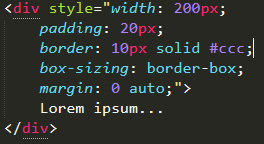
 

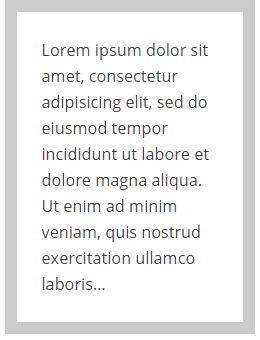
Como se puede comprobar, no mide 200px de ancho, sino 260px. Es decir: 200px de ancho inicial, más 20px de padding izquierdo, más 20px de padding derecho, más 10px de borde izquierdo, más 10px de borde derecho.

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:

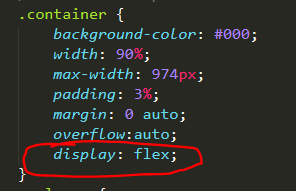


Ahora ese elemento exactamente 200px, ni uno más, ni uno menos.

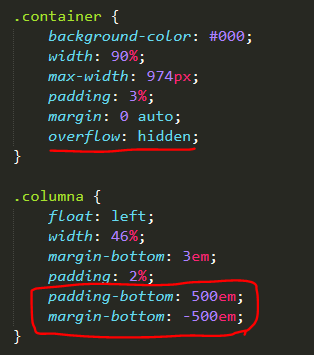
Esto es muy útil para elementos fluídos, cuando necesitas que el elemento ocupe (por ejemplo) un 33% del ancho, y si ocupa un píxel más toda la estructura se  estropearía.

### 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 soluciones es añadir la propiedad “display: flex” al container. Haciendo que todo lo que este dentro tenga la misma altura.



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 contenerdor tenga “overflow: hidden”



## Pregunta 3. Sobre diseño web adaptativo, responde a las siguientes cuestiones:

### ¿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).

### ¿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.