Lenguaje de Marcas y Sistemas de Gestión

UA 2.14 - Introducción CSS: Maquetación Web

Centro Profesional
Universidad
Europea

Raúl Rodríguez Mercado
raul.rodriguez@universidadeuropea.es / @raulrodriguezue
Dpto. Ciencias y Tecnología de la Informática y Comunicación

UA 2.14: Introducción CSS: Maquetación Web

Objetivos

- Definir qué es CSS

- Conocer la estructura de las hojas de estilo y como se aplican a los documentos HTML

UA 2.14: Introducción CSS: Maquetación Web

Contenidos

- Definir la estructura de una página web.
- Conocer el comportamiento de las distintas etiquetas al colocarlas en una página web.
- Comprender cómo fluyen los elementos en la pantalla del navegador.
- Utilizar para todo ello la forma “tradicional”.

UA 2.14: Introducción CSS: Maquetación Web

¿Qué es la Maquetación Web?

- La **maquetación web** es la parte que se encarga de definir la **disposición** de los distintos elementos (imágenes, texto etc…) que forman nuestra página web para definir la **estructura visual** de la misma de acuerdo a unos objetivos de comunicación definidos.

¿Cómo se va a conseguir esta disposición?

CSS +

ETIQUETAS

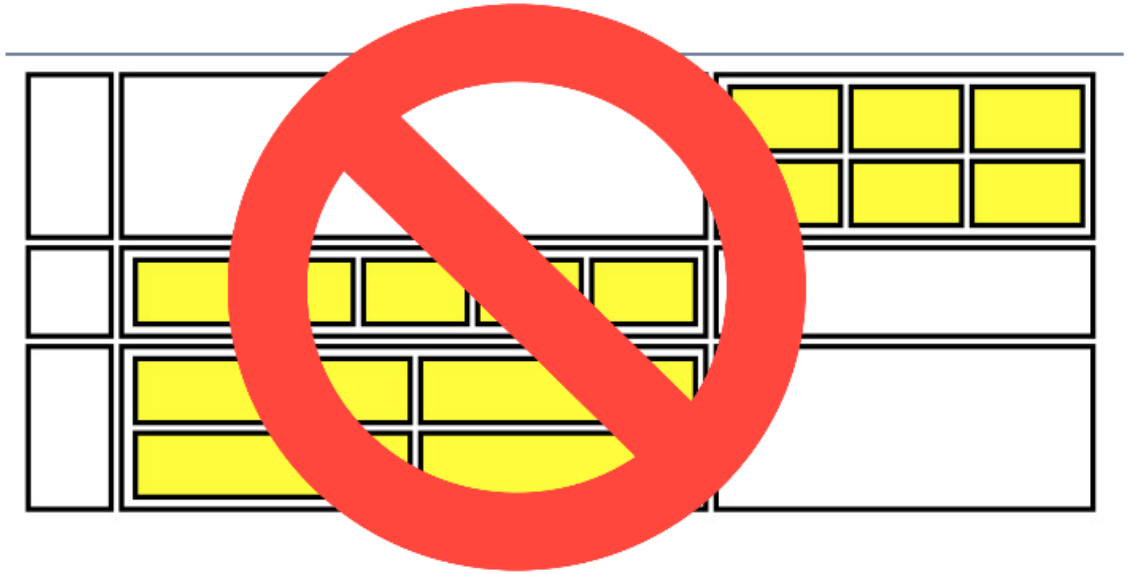
div, header,footer,nav, section,article, aside, dialog, main, summary…

Se utilizan las etiquetas DIV y además las nuevas etiquetas existentes en HTML5 semánticas (section, aside, etc.) que son iguales que los divs pero que facilitan a los navegadores a la hora de identificarlas y saber qué tipo de información es la que llevan dentro.

UA 2.14: Introducción CSS: Maquetación Web

¿Por qué no utilizar las tablas?

- No hay separación de apariencia y contenido.
- Mucho esfuerzo si hay cambio de diseño.
- Problemas para hacer páginas responsivas.
- Son menos accesibles para los dispositivos.
- Se cargan de manera más lenta.



- Son menos legibles (etiquetas semánticas) y tardan más en cargarse
- Son “peores” para el tema de posicionamiento SEO

UA 2.14: Introducción CSS: Maquetación Web

Ejemplo Maquetación Web

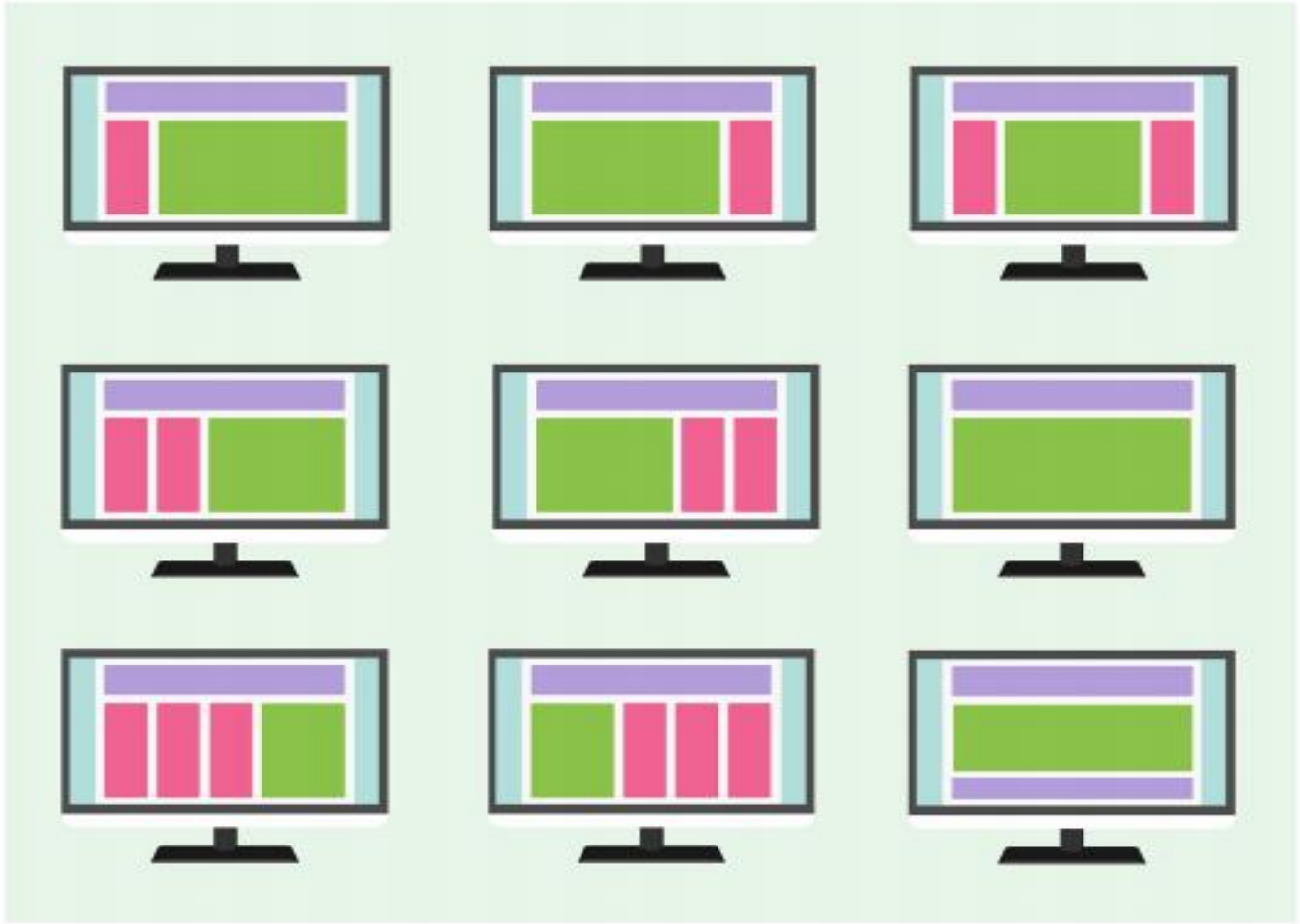


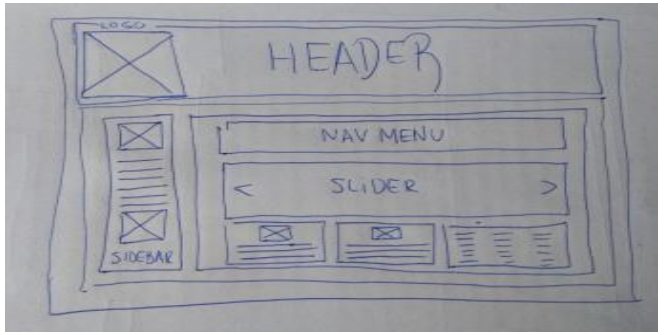
Image under CC0 from https://pixabay.com/es/users/JuralMin-2051452/

UA 2.14: Introducción CSS: Maquetación Web

Recomendaciones para Maquetar Correctamente

1. ESQUEMA PREVIO

- En esta, como en otras muchas actividades, es bueno pararse a pensar antes de lanzarse a desarrollar. Si inviertes un poco de tiempo en pensar tu diseño de antemano al final seguro que ahorras tiempo.
- Para esto lo mejor es realizar bocetos de nuestro diseño. Estos bocetos podremos hacerlo principalmente de dos maneras:
  - ✓ ***A mano*** que es la forma más fácil y rápida.
  - ✓ Con ***herramientas*** que aunque no son tan rápidas me van a permitir operaciones como importar, compartir etc… Algunas herramientas para realizar estos bocetos son:
    - mockflow.com
    - moqups.com
    - wireframe.cc
    - balsamiq.com
    - ninjamock



UA 2.14: Introducción CSS: Maquetación Web

Recomendaciones para Maquetar Correctamente

2. **“DE LO GRANDE A LO MÁS PEQUEÑO”**

- No pretendéis poner la imagen primero, sin haber puesto antes el elemento en el que va a estar alojado.
- No coloco lo que hay dentro de una zona del diseño hasta que no he colocado el contenedor principal, el contenedor padre.

3. **“USAR LAS AYUDAS DISPONIBLES”**

- Todas los navegadores tienen herramientas disponibles que nos van a ayudar a la hora de maquetar y diseñar nuestra web. Estos están disponibles cuando por ejemplo visualizamos la opción de “inspeccionar elementos”.

4. **“COPIAR”**

- Hay millones de webs. Si ves alguna que te guste usa las herramientas de los navegadores e investiga cómo lo han hecho. Utiliza el conocimiento de otras personas en tú favor.
- Pero ojo, no te limites a copiar y pegar. Así no aprenderás. Entiende lo que están haciendo y usa la documentación técnica si hay partes que no ves claras.

Recomendaciones para Maquetar Correctamente

5. “PROBAR EN DISTINTOS NAVEGADORES”

- Los navegadores tienen sus propias hojas de estilos por defecto que pueden hacer que la misma página no sea vea igual en todos los navegadores.
- Conforme vayas acabando tus diseños deberás comprobar que todo se ve bien en los distintos navegadores. Hacer esto sobre todo al principio. Posteriormente tus diseños serán más robustos y no será tan necesario.

UA 2.14: Introducción CSS: Maquetación Web

Recomendaciones para Maquetar Correctamente

6. “PACIENCIA… MUCHA”

- La maquetación Web no es difícil pero requiere la comprensión de ciertas reglas básicas y su perfecto encaje posterior.

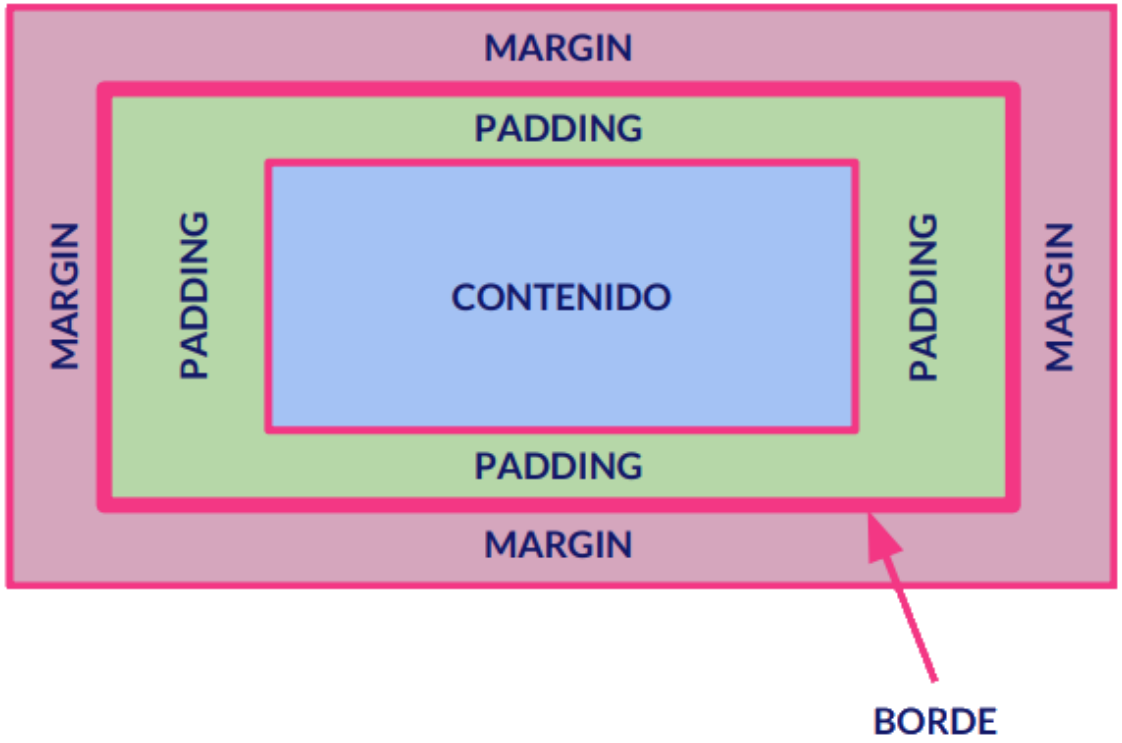
- En tus inicios muchas veces desesperarás y pensarás que es el navegador el que funciona mal... No es así, si el navegador no muestra lo que tú quieres es porque está mostrando lo que tú le has dicho. Si eso sucede:

  - ✓ Revisa los conceptos básicos.
  - ✓ Revisa la sintaxis de tu web.
  - ✓ Revisar los estilos por defecto.
  - ✓ Vuelve al uno hasta que todo esté bien.
  - ✓ Es decir…***TEN PACIENCIA***

Flujo de Elementos en la ventana de nuestro navegador

Consideraciones Importantes

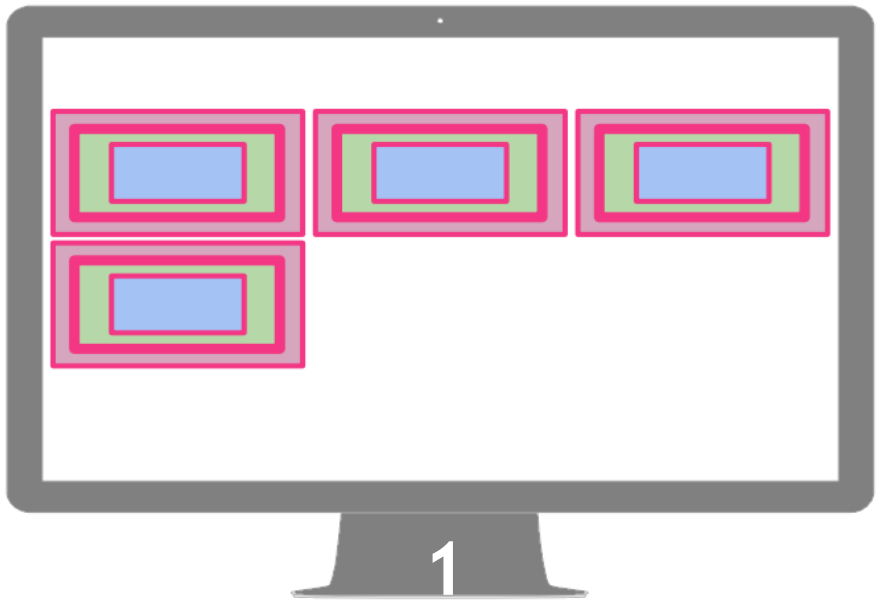
- Recordar que todas las etiquetas que se van a representar son *cajas*.
- Los navegadores no hacen **NADA** para controlar el diseño de nuestra página Web.
- Los navegadores, lo único que hacen es mostrar los elementos de nuestra página HTML en el mismo **ORDEN** en el que los hemos escrito.

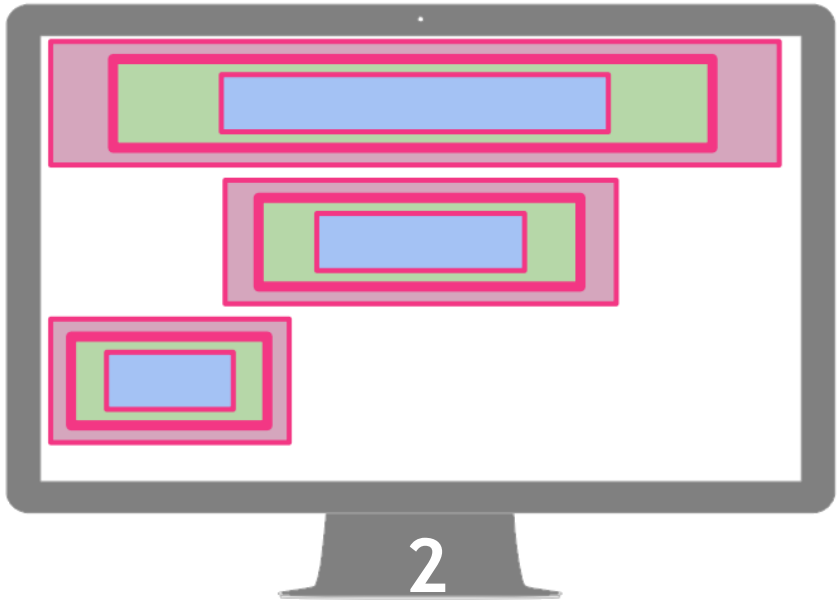


Flujo de Elementos en la ventana de nuestro navegador

Consideraciones Importantes

- Siguen solo dos reglas básicas dependiendo de las propiedades de las cajas:
  - ✓ Determinados tipos de caja se van poniendo unas detrás de otros mientras quepan en la pantalla (1)
  - ✓ Cuando no caben las cajas pasan a la “siguiente línea” del navegador. (2)
- Otros tipos de caja provocan un “salto de línea”.





Flujo de Elementos en la ventana de nuestro navegador

¿Y Entonces?

ENTONCES...

DISEÑO = CSS

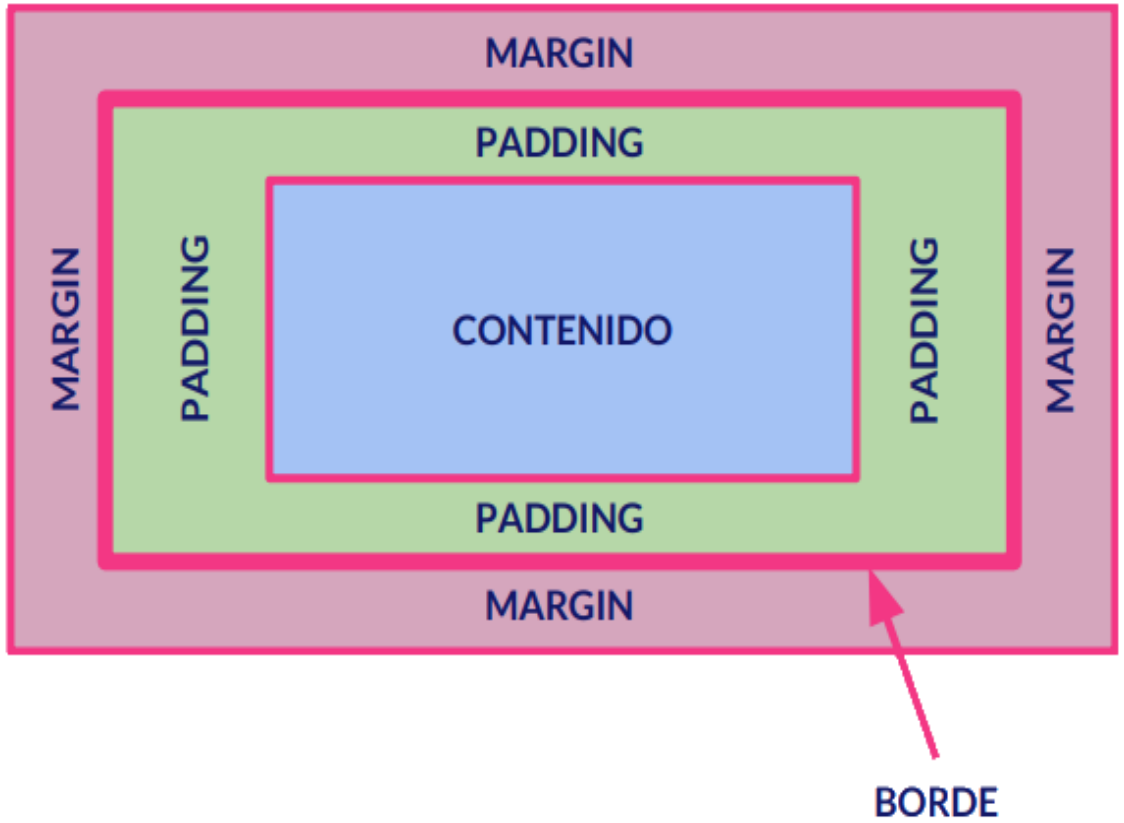
Todo lo tenemos que hacer nosotros. Y recordad que hay estilos por defecto.

UA 2.14: Introducción CSS: Maquetación Web

En Línea y en Bloque: Propiedad Display

Recordar…

- Pese a que todos los elementos de mi página HTML son cajas lo cierto es que no todas las cajas se comportan igual cuando las añadimos.



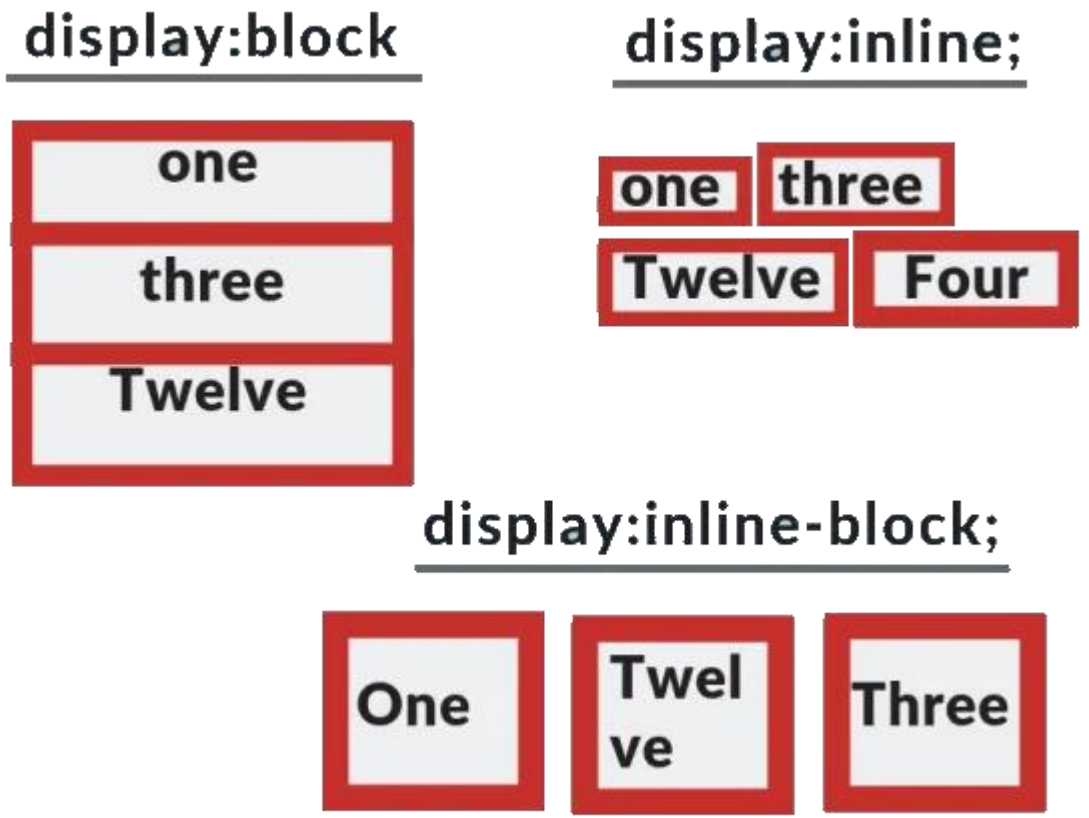
Su comportamiento viene determinado por la propiedad **display**

UA 2.14: Introducción CSS: Maquetación Web

En Línea y en Bloque: Propiedad Display

Valores de Display más usados

- ✓ inline
- ✓ inline-bock
- ✓ block
- ✓ none
- ✓ Relacionados con tablas
- ✓ Flex y Grid → Veremos más adelante



Y algunos más: https://developer.mozilla.org/es/docs/Web/CSS/display

En Línea y en Bloque: Propiedad Display

INLINE, INLINE-BLOCK

Inline

Los elementos Inline no rompen el flujo de la línea y se van colocando uno detrás de otro mientras caben. Aceptan *margin y padding* (solo sirve en horizontal). Ignoran *width y height*.

Inline-Block

Como inline pero podemos asignarles *width y height*.

,<a>,<b>,<img>...

En Línea y en Bloque: Propiedad Display

DISPLAY:BLOCK

BLOCK

Los elementos en bloque rompen el flujo de la línea y provocan "una salto de línea" tanto anterior como posterior. Por defecto si no especificamos una anchura ocupan toda la del elemento que los contiene (la etiqueta padre)

<h1>,<p>,<section>, ,<ul>,<li>,<nav>...

En Línea y en Bloque: Propiedad Display

DISPLAY:NONE

DISPLAY:NONE

NONE

Una vez fijada el elemento desaparece. No deja dejará un espacio vacío aunque siga en el código HTML. La propiedad visibility:hidden sí que deja el hueco aunque no se muestre.

En Línea y en Bloque: Propiedad Display

RELATIVOS A TABLAS

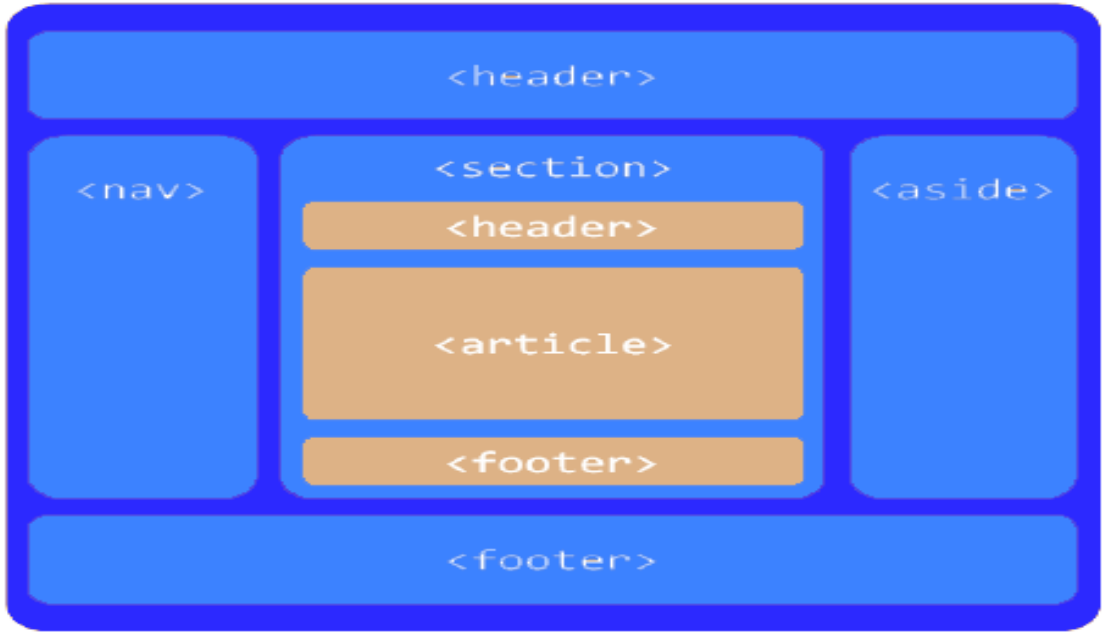
- Emulan el comportamiento de una tabla

VALORES DE DISPLAY RELACIONADOS CON TABLAS

- table
- table-cell
- table-row
- table-caption
- table-column
- table-colgroup
- table-header-group
- table-footer-group
- table-row-group

Layout en HTML y CSS

- El ***layout*** **o plantilla es un esquema de la distribución de los elementos dentro de una página web.**
- Se compone de una serie de bloques de ciertas dimensiones en los que se colocará el contenido.
- Estos bloques suelen trazarse a través de etiquetas HTML comunes, como **div**, o **semánticas, como header, nav, section, article, aside y footer,** incorporadas en HTML5.



Tipos de Layout

Definición

- Los tipos de Layout que nos podemos encontrar son los siguientes:

  - Fixed
  - Elastic
  - Fluid (%)
  - Con Min/Max Sizing
  - Responsive
  - Mezclas de varios.

UA 2.14: Introducción CSS: Maquetación Web

Tipos de Layout

FIXED

- El tamaño (anchura) de todos los elementos se establece con pixels.

VENTAJA

- ✓ Siempre el mismo tamaño
- ✓ Total control

DESVENTAJA

- ✓ Pantallas pequeñas -> Scroll Horizontal
- ✓ Pantalla grandes -> Mucho espacio en blanco a los lados si el contenedor principal no es muy ancho.

Tipos de Layout

ELASTIC

- El tamaño (anchura) de todos los elementos se establece con em

VENTAJA

- ✓ Tamaño en em escala correctamente

DESVENTAJA

- ✓ Elementos adyacentes pueden solaparse
- ✓ Habría que hacer pruebas en todo tipo de dispositivos y con todo tipo de tamaños de letra

Tipos de Layout

FLUID

- El tamaño (anchura) de todos los elementos se establece con % (siempre con respecto a la etiqueta padre)

VENTAJA

- ✓ La proporción de los elementos es siempre la misma

DESVENTAJA

- ✓ En pantallas pequeñas las columnas puede ser muy estrechas.
- ✓ En columnas estrechas texto largos -> celdas muy altas.
- ✓ Problemas si hay imágenes y vídeos :(

Tipos de Layout

MAX/MIN WIDTH (HÍBRIDO)

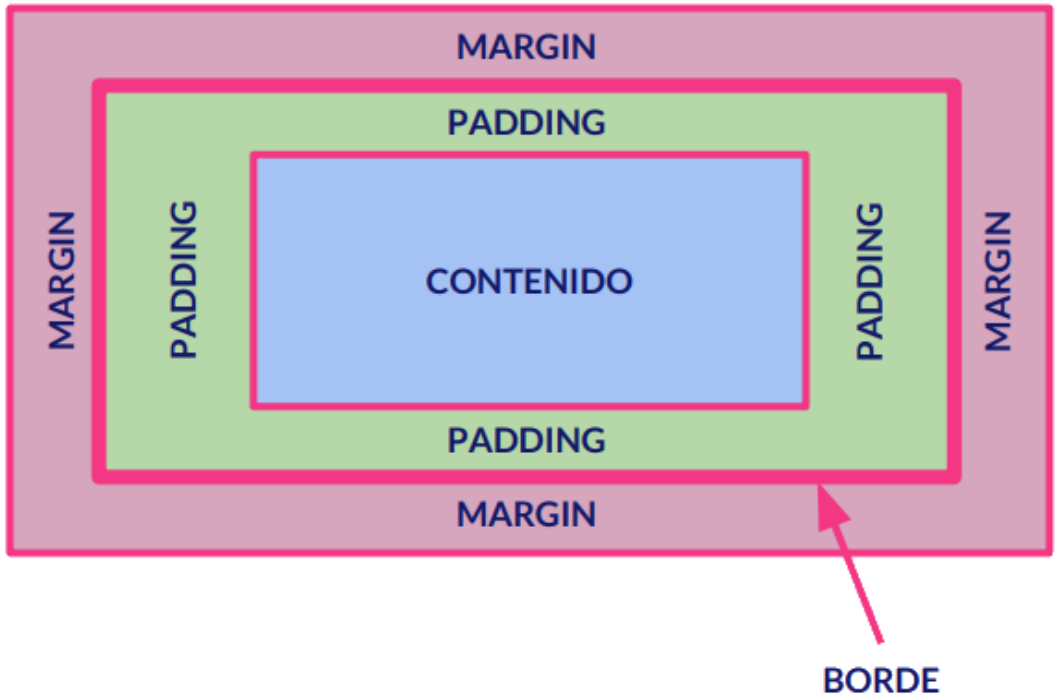
- El tamaño (anchura) puede crecer/encoger con unos límites max-width / min-width expresados en pixels.

RESPONSIVE

- Layout que cambia dependiendo de las características de la pantalla, normalmente dependiendo sobre todo de la anchura de la pantalla (sólo uno, cambio fluido) Si tengo más de un layout para la página se denomina adaptative (cambio brusco)

Box-sizing

Recordemos: El modelo de la caja suele ser de esta manera:

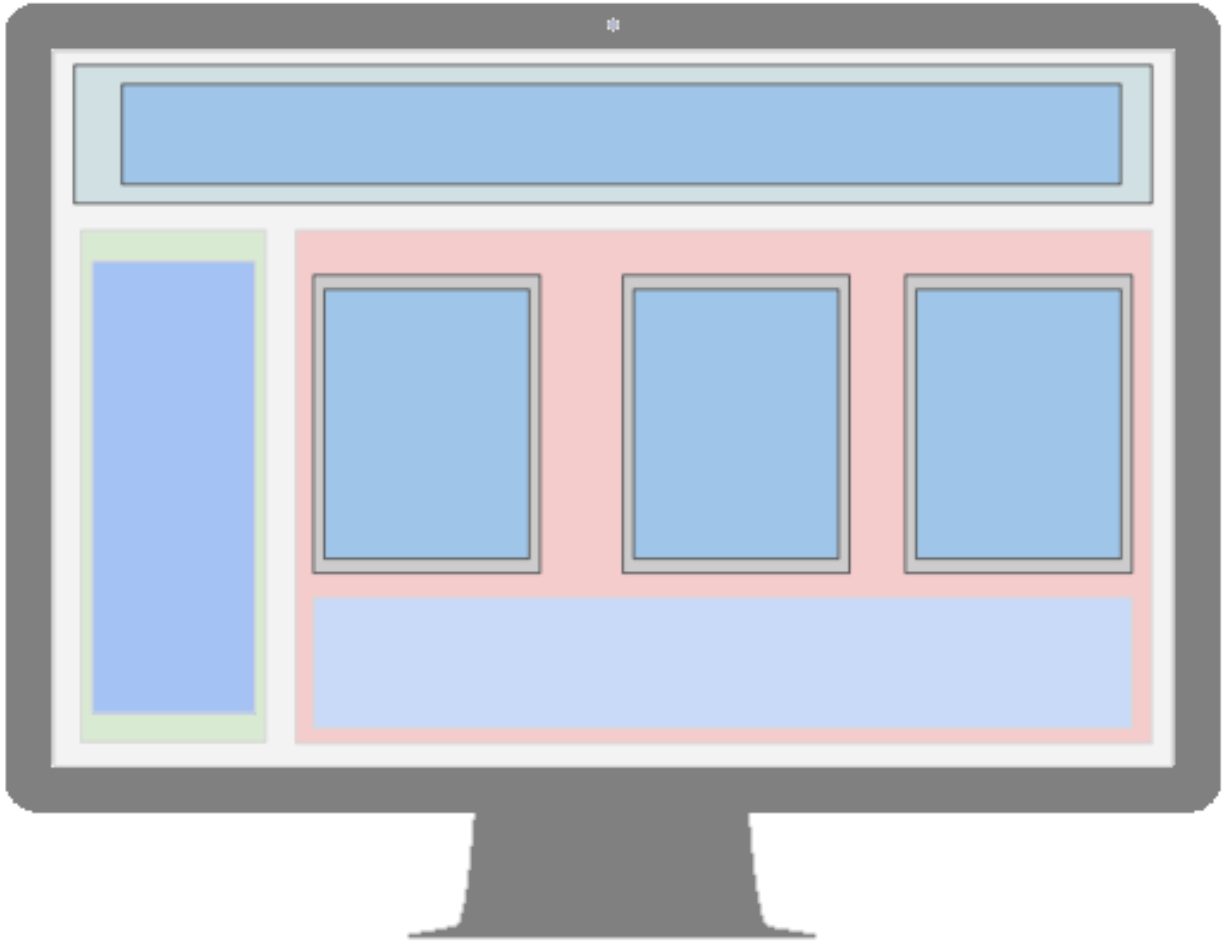


- ✓ **Contenido** es la zona donde va el elemento propiamente dicho.
- ✓ **Padding** es la distancia entre el contenido y el borde.
- ✓ El **Borde** es el límite entre el elemento y el resto de los elementos.
- ✓ EL **Margin** es la separación entre el elemento y los demás elementos.

UA 2.14: Introducción CSS: Maquetación Web

Box-sizing

- **Recordemos…:** Maquetación web consiste en disponer estas cajas para que cada una ocupa el lugar que queremos al ser mostradas en nuestro navegador



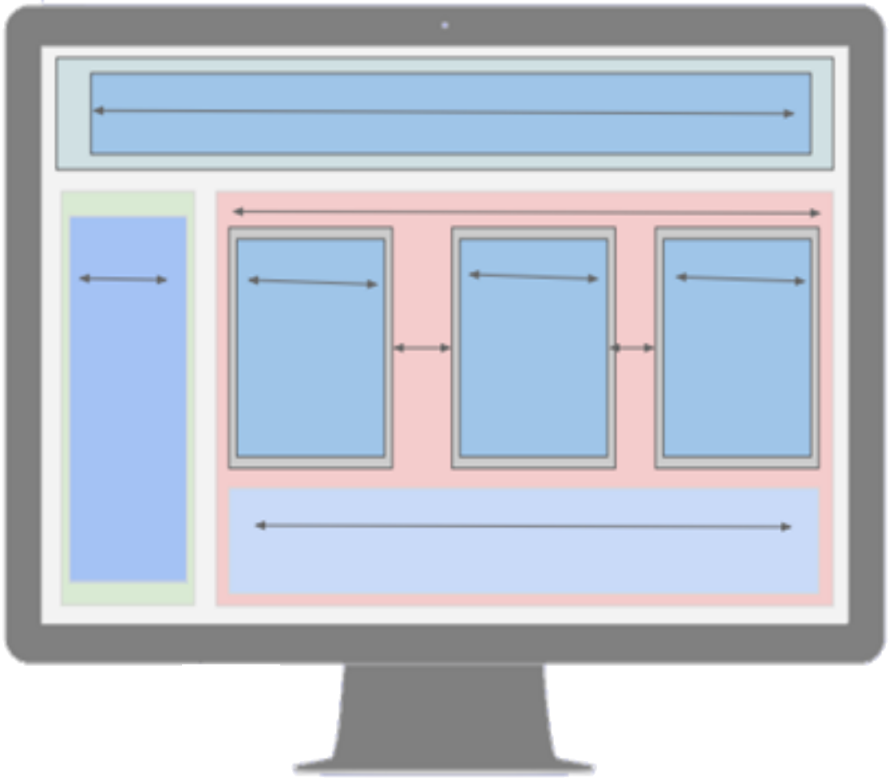
Box-sizing

- Por defecto, los navegadores calculan las dimensiones de los elementos de la siguiente manera:

Altura del elemento = altura del contenido+ padding + borde

y

Anchura del elemento = anchura del contenido+ padding + borde



Box-sizing

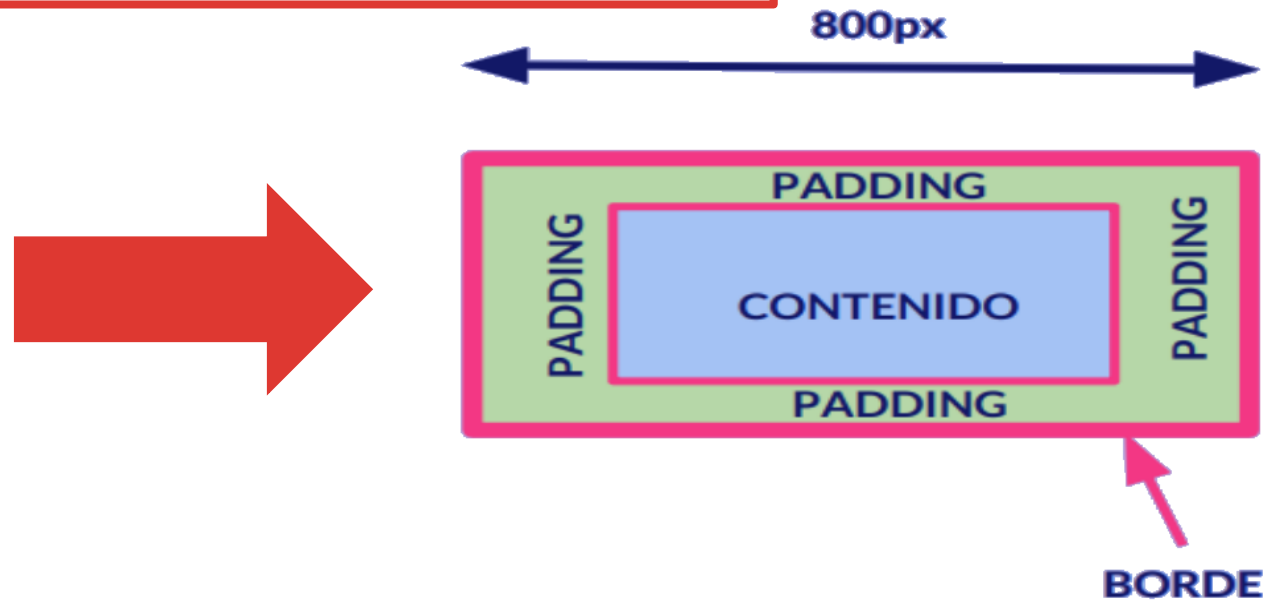
- Para que nos resulte más fácil estructurar todo y “echar menos cuentas”, lo que podemos poner en nuestro CSS es la siguiente estructura:

```
* {
 -webkit-box-sizing: border-box;
 -moz-box-sizing: border-box;
 box-sizing: border-box;
 }
```

UN VALOR
BORDER-BOX

En la anchura que le doy ya se incluye contenido, padding y border. No tengo que hacer más cálculos.

```
* {
   width: 800px;
}
```

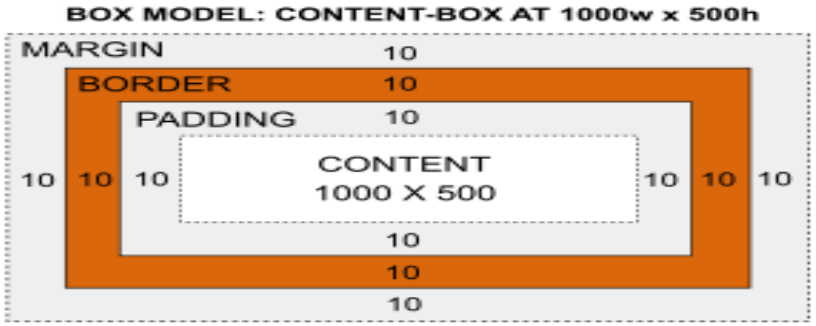


Box-sizing

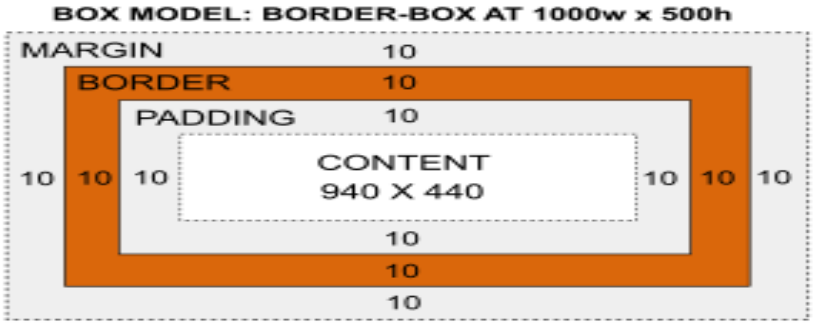
- La propiedad ***box-sizing*** puede tener otros dos valores:

  - ✓ ***content-box*** que es el funcionamiento por defecto, para saber que ocupa la caja debo sumar todo.

  - ✓ ***padding-box*** que no tienen en cuenta el borde pero si el padding y el contenido. Aún no tiene soporte en la gran mayoría de los navegadores a pesar de que sí lo reconocen.




```
div{
    width: 1000px;
    height: 500px;
    margin: 10px;
    border: 10px solid orange;
    padding: 10px;
    box-sizing: content-box;
}
```




```
div{
    width: 1000px;
    height: 500px;
    margin: 10px;
    border: 10px solid orange;
    padding: 10px;
    box-sizing: border-box;
}
```

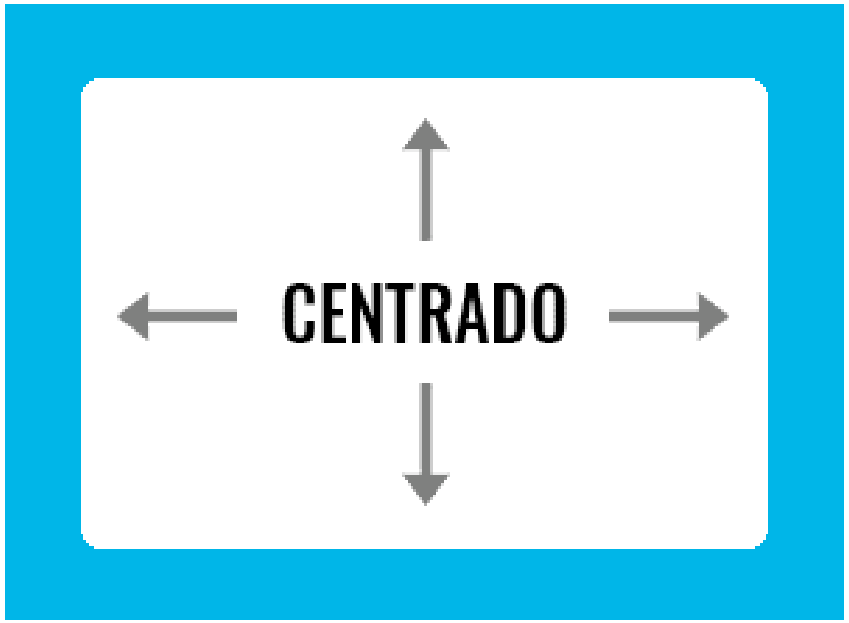
UA 2.14: Introducción CSS: Maquetación Web

Cómo centrar un elemento

- Cuando hablamos de centrar un elemento, el contexto de referencia es siempre su etiqueta padre o etiqueta contenedora.

¿ES DIFÍCIL?

Suele ser un concepto que da muchos dolores de cabeza a los iniciados. Vamos intentar dejarlo todo muy muy claro.



Cómo centrar un elemento

CENTRADO HORIZONTAL

- Elementos en Línea: ***text-align: center*** (***al padre***)
- Elementos en Bloque:
  - ✓ ***margin: X auto; (al elemento. Debe de tener anchura)***
- Varios elementos en Bloque en la misma fila:
  - ✓ ***text-align: center (al padre)***
  - ✓ ***display: inline-block (a los elementos)***
  - ✓ ***Deben tener anchura***
- Con contenedores flex

Cómo centrar un elemento

CENTRADO VERTICAL: ELEMENTOS EN LÍNEA

- El **mismo padding** arriba y abajo
- **vertical-align:middle** si dentro de elemento de tabla o lo estamos simulando con propiedad display. (el padre debe tener altura fija)
- Con contenedores flex

CENTRADO VERTICAL: ELEMENTOS EN BLOQUE

- Utilizando la **propiedad position en el contenedor y en el elemento**
- Con contenedores flex

Propiedad Position: Posicionando Elementos

- Con la propiedad ***position*** podremos "romper" el flujo normal de los elementos en cuanto a su posicionamiento natural dentro de nuestras páginas web, y ponerlos en otro sitio. Se utiliza para maquetar de una manera más fina y correcta los elementos y contenidos.

- Los valores que puede tener son los siguientes que se comentarán más detalladamente:

  - ✓ **static**
  - ✓ **relative**
  - ✓ **absolute**
  - ✓ **fixed**
  - ✓ **sticky**
  - ✓ **inherit**

- Estos valores van unidos a las propiedades top,bottom,left,right (desplazamiento) y z-index (capas).

UA 2.14: Introducción CSS: Maquetación Web

Propiedad Position: Posicionando Elementos

EXPLICACIÓN RÁPIDA

static

Es el valor por **defecto**.

El elemento sigue el flujo que le corresponde. Aunque use top, bottom, left,right o z-index **NO** las aplica.

relative

Como static pero **SÍ** atiende top, bottom, left, right o z-index a partir de la posición que le corresponde por el flujo.

fixed

Se le aplica top, bottom, left, right o z-index **en relación al documento**. **No** atiende al **scroll**. Permanece siempre en el mismo sitio.

absolute

Se comporta como **fixed** pero en relación a la primera etiqueta antecesora que tenga **position: relative.**

sticky

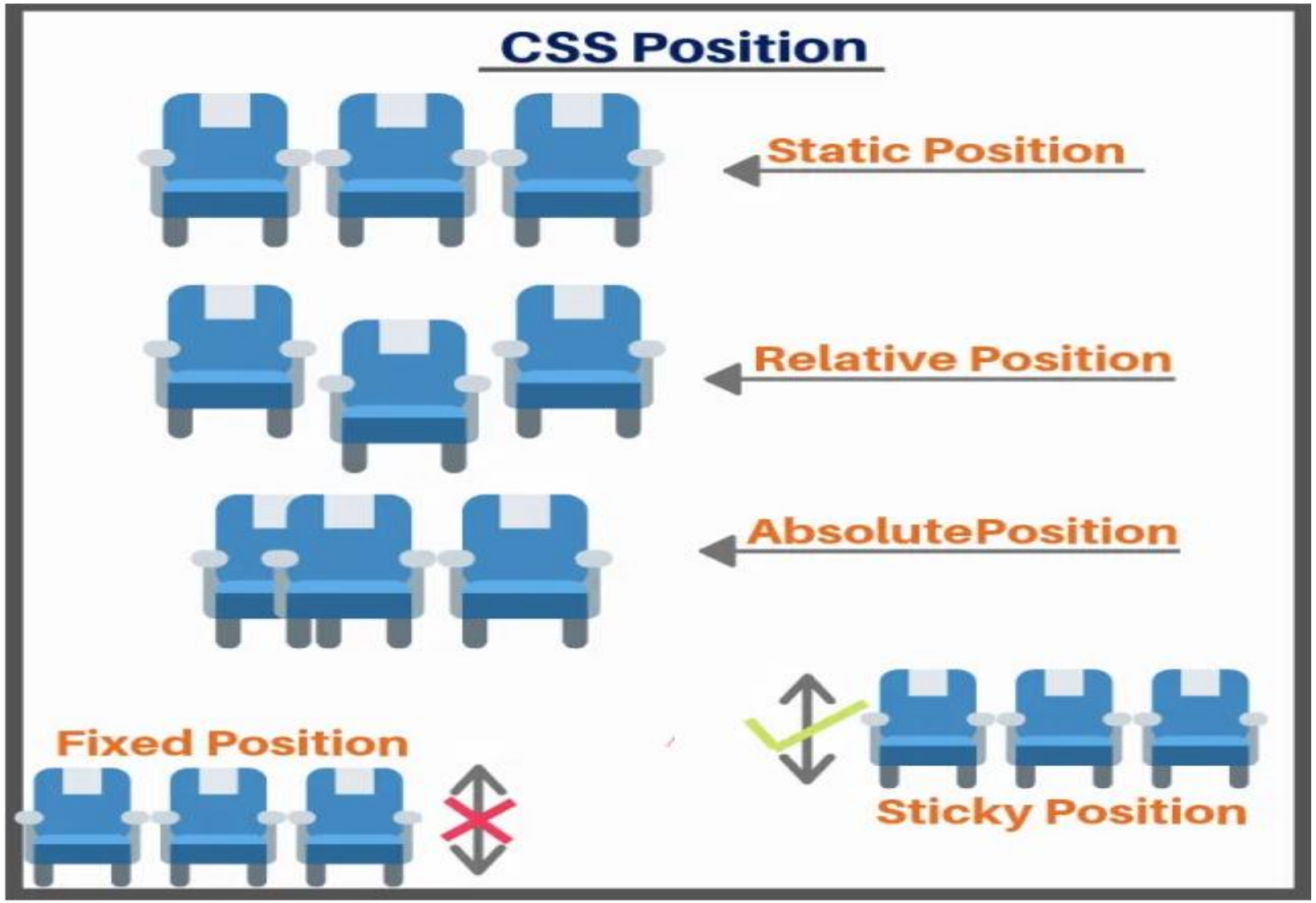
relative hasta llegar a una posición de scroll y a partir de entonces **fixed.**

inherit

La propiedad **position** no se propaga en cascada, Si queremos que sea así añadiremos el valor **inherit** a los hijos que queremos que hereden

UA 2.14: Introducción CSS: Maquetación Web

Propiedad Position: Posicionando Elementos

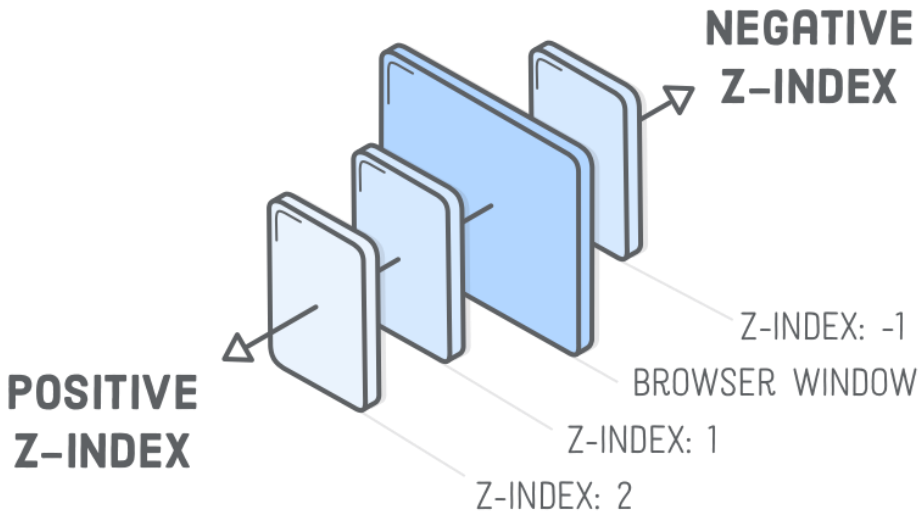


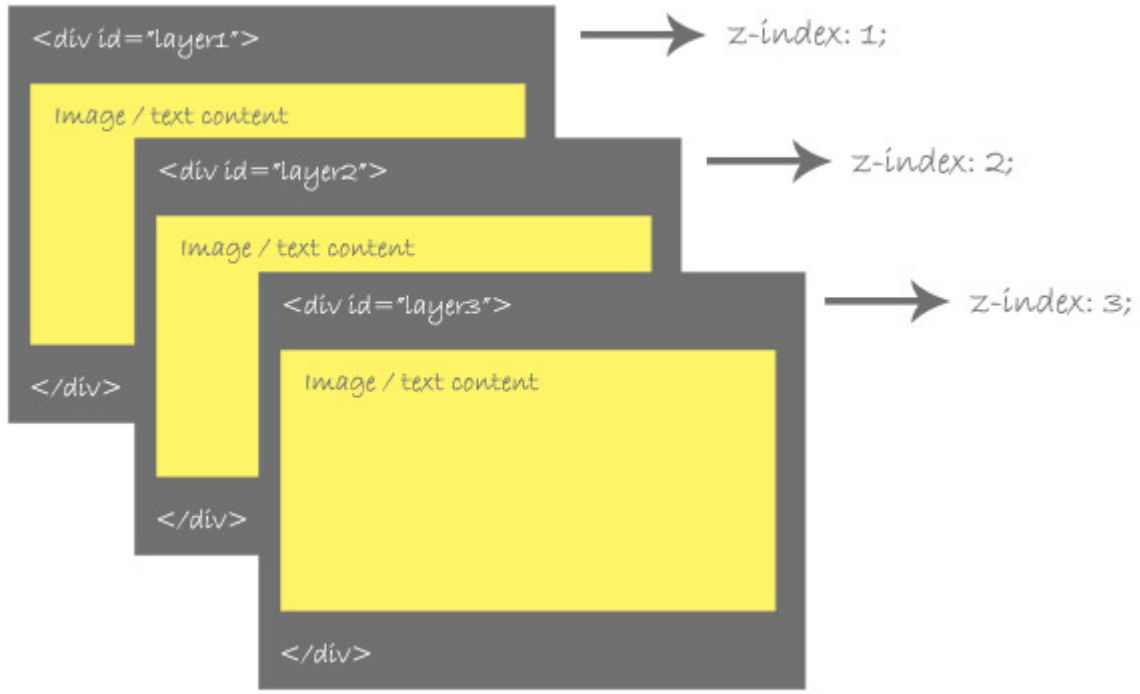
UA 2.14: Introducción CSS: Maquetación Web

Propiedad Position: Posicionando Elementos

Z-Index

- Si al ***POSICIONAR*** elementos con se "***solapan***" y ocupan una misma área con ***z-index*** podemos establecer el ***orden***. Podemos establecer "***capas***".





UA 2.14: Introducción CSS: Maquetación Web

Propiedad Position: Posicionando Elementos

CENTRADO VERTICAL CON POSITION (para elementos de bloque)

Si conocemos la altura del elemento y esta es por ejemplo 150px;

```
.contenedor {
        position: relative;
}

.elemento_a_centrar {
        height: 150px;
        margin-top: -75px; /** La mitad de la altura **/
        position: absolute;
        top: 50%;
}
```

No conocemos la altura

```
.contenedor {
        position: relative;
}

.elemento_a_centrar {
        position: absolute;
        top: 50%;
        transform: translateY(-50%);
}`
```

Columnas en CSS

- En CSS podemos dividir el contenido que queremos mostrar en varias columnas tal y como podemos ver en un periódico.

Propiedades para Columnas en CSS

- ✓ **column-count**: nº de columnas indicadas al contenedor padre.
- ✓ **column-width**: para fijar el ancho de las columnas.
- ✓ **column-gap:** separación entre columnas
- ✓ **column-rule:** estilo para la línea que separa las columnas (como border)
- ✓ **column-span:** si el elemento sigue el número de columnas o no (valores ***all y none***)
- ✓ **column-fill:** para establecer cómo se rellenan las columnas. El contenedor debe tener altura. Valores ***auto*** o ***balance*** (todas las columnas la misma altura)
- ✓ **break-inside: avoid** si queremos que el elemento no quede roto de una columna a otra.

Columnas en CSS

- En CSS podemos dividir el contenido que queremos mostrar en varias columnas tal y como podemos ver en un periódico.

Propiedades para Columnas en CSS

- ✓ **column-count**: nº de columnas indicadas al contenedor padre.
- ✓ **column-width**: para fijar el ancho de las columnas.
- ✓ **column-gap:** separación entre columnas
- ✓ **column-rule:** estilo para la línea que separa las columnas (como border)
- ✓ **column-span:** si el elemento sigue el número de columnas o no (valores ***all y none***)
- ✓ **column-fill:** para establecer cómo se rellenan las columnas. El contenedor debe tener altura. Valores ***auto*** o ***balance*** (todas las columnas la misma altura)
- ✓ **break-inside: avoid** si queremos que el elemento no quede roto de una columna a otra.

Elementos Flotantes

PROPIEDAD FLOAT

- La propiedad **float (left o right)** está pensada para especificar cómo se dispone un texto alrededor de una imagen...

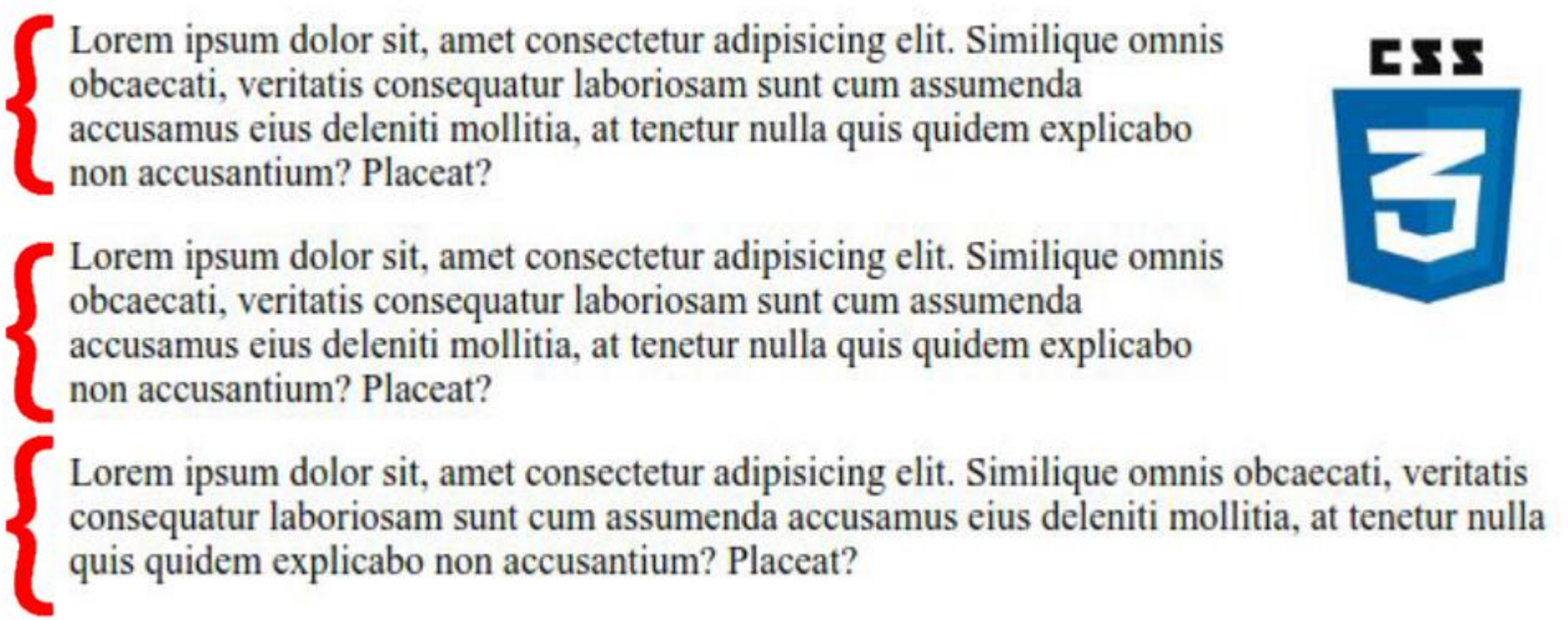
```
img {
    float: right;
     margin: 0px
1em;
}
```

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Similique omnis obcaecati, veritatis consequatur laboriosam sunt cum assumenda accusamus eius deleniti mollitia, at tenetur nulla quis quidem explicabo non accusantium? Placeat?

Elementos Flotantes

PROPIEDAD FLOAT

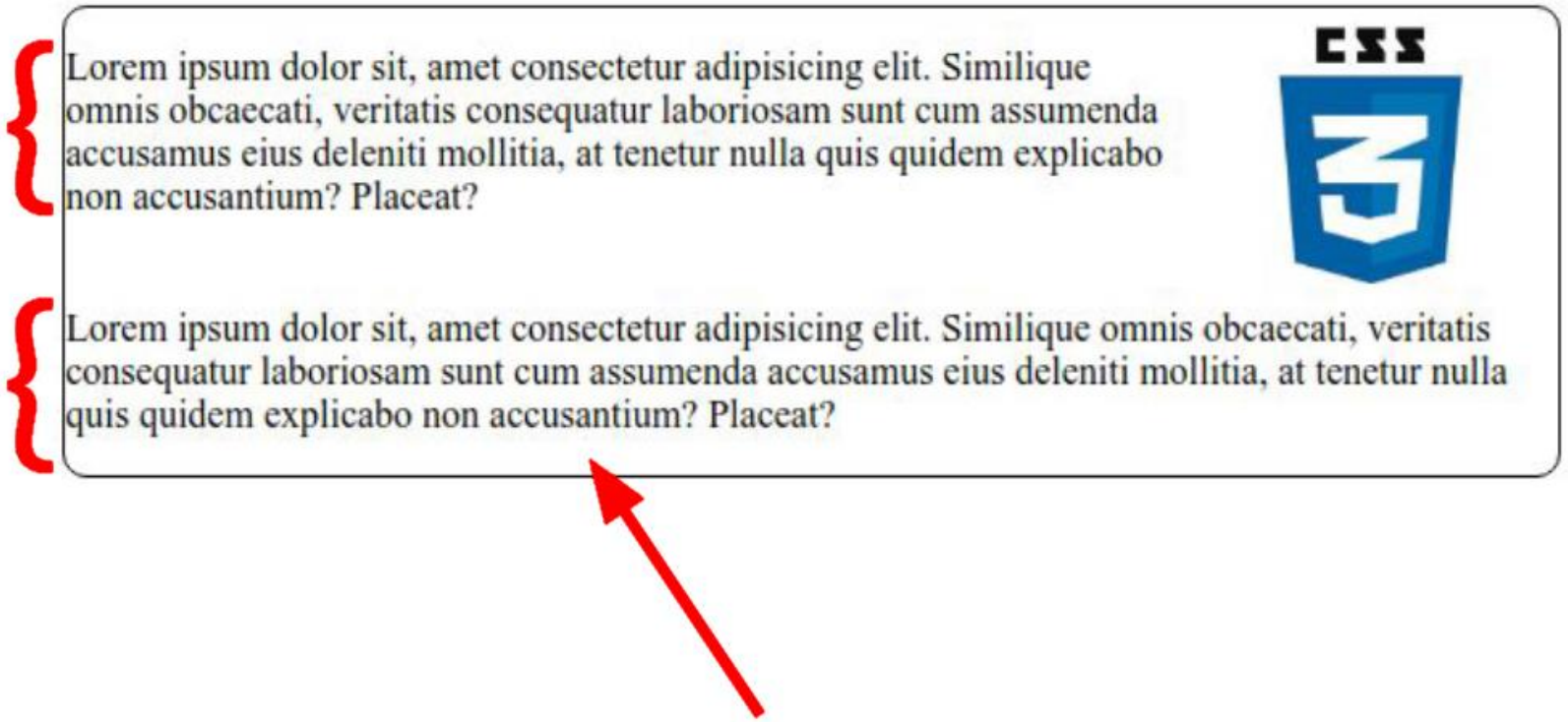
- Si sigue habiendo ***"hueco vertical"*** en el lugar contrario al que he flotado la imagen los elementos se siguen añadiendo ahí hasta que sobrepasan en la ***"vertical"*** al elemento flotante.



Elementos Flotantes

PROPIEDAD CLEAR

- Si queremos ***"forzar"*** que un elemento deje de flotar, debemos añadir la propiedad ***clear:right*** **(o left o both)** y ese elemento ya se añadirá tras el fin en vertical del elemento flotante...



UA 2.14: Introducción CSS: Maquetación Web

Elementos Flotantes

CLEARFIX HACK

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Similique omnis obcaecati, veritatis consequatur laboriosam sunt cum assumenda accusamus eius deleniti mollitia, at tenetur nulla quis quidem explicabo non accusantium? Placeat?

La soluciones son dos (ambas son propiedades CSS al elemento contenedor)

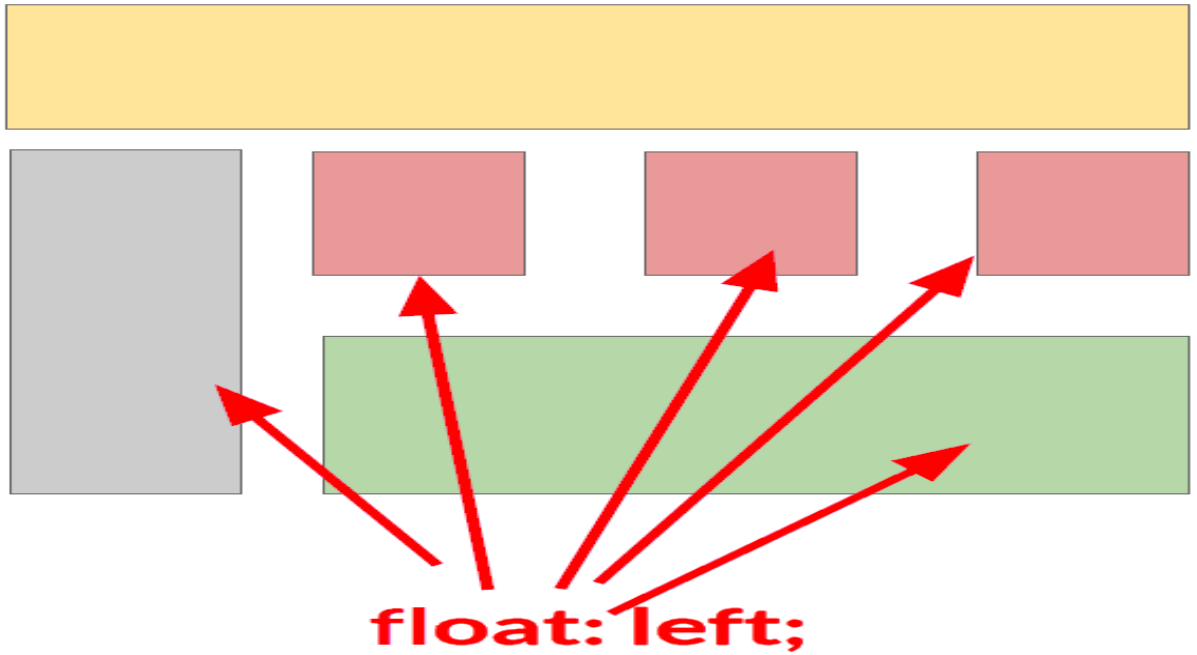
```
selector {
 overflow-y: auto;
 height: Altura_suficiente; /*Una de las dos*/
}
```

Elementos Flotantes

CREANDO LAYOUTS

- Posteriormente los desarrolladores se dieron cuenta de que los elementos flotantes se podían usar para maquetar. Lo conseguiremos:

  ✓ **Flotando los elementos según necesitemos** (div, nav, header…)
  ✓ **Dándoles las dimensiones** adecuadas.




Universidad
Europea