PRINCIPIOS DE MAQUETACIÓN CON CSS

1.

¿ Qué es lo que voy a aprender?

EN ESTE CURSO

- Definiré la **estructura** de mi página web.
- Conoceré el **comportamiento** de las distintas etiquetas al colocarlas en mi página.
- Comprenderé cómo **fluyen** los elementos en la pantalla del navegador.
- Utilizaré para todo ello la **forma “tradicional”.**

2.

¿ Qué es lo que NO voy a aprender?

EN ESTE CURSO NO...

- Haré **Diseño Responsivo.**
- Crearé **Efectos y animaciones** CSS3.
- Utilizaré contenedores **flex y grid** (la nueva forma de maquetar con CSS3).

3.

Requisitos


HTML
5


CSS
3

¿ Qué es la MAQUETACIÓN WEB?

1.

DEFINICIÓN

Una base formal

“

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.

2.

¿CÓMO?

Para conseguir
esa disposición

CSS +

ETIQUETAS

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

3.

¿TABLAS?

Yo ya sé tablas

¿POR QUÉ NO TABLAS ?

- No hay separación de apariencia y contenido.
- Mucho esfuerzo si hay cambio de diseño.
- Problemas para hacer páginas responsivas.
- Son menos accesibles.
- Se cargan de manera más lenta.
- Son menos legibles (etiquetas semánticas)
- Pero SEO (no puedo demostrarlo…)

4.

UN EJEMPLO

Vamos a ver

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

ENTORNO DE TRABAJO PARA EL CURSO

ALGUNAS EXTENSIONES PARA VSCODE

- Auto Close Tag
- Auto Rename Tag
- GitLens
- Intellisense for CSS
- Prettier - Code Formatter

REFERENCIAS!

https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout

ESTRUCTURA DEL CURSO

1.

PRESENTACIÓN DEL PROYECTO

2.

RECOMENDACIONES

3.

TIPOS DE ELEMENTOS

4.

FLUJO DE LOS ELEMENTOS

5.

TIPOS DE LAYOUTS

6.

POSICIONANDO ELEMENTOS

7.

COLUMNAS

8.

ELEMENTOS FLOTANTES

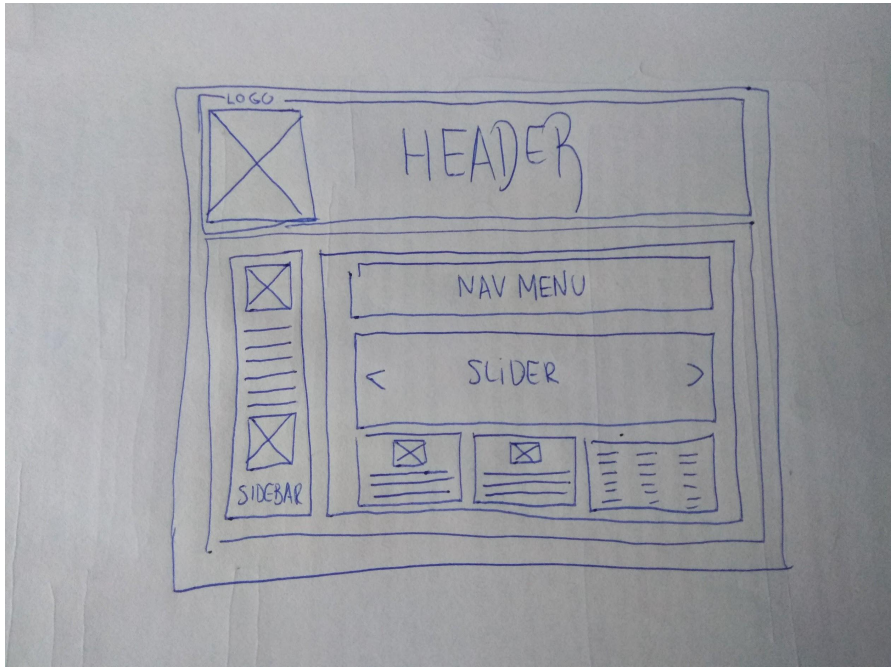
ALGUNAS RECOMENDACIONES

1.

ESQUEMA PREVIO

Te ahorrará
mucho tiempo

HECHO A MANO



HERRAMIENTAS ONLINE

- mockflow.com
- moqups.com
- wireframe.cc
- balsamiq.com
- ninjamock.com

2.

“DE LO GRANDE A LO CHICO”

No empieces la casa por el tejado

“

No coloco lo que hay dentro de una zona del diseño hasta que no he colocado el contenedor principal, el contenedor padre.

3.

USO LAS AYUDAS DISPONIBLES

Todo los navegadores las llevan

4.

COPIA

Es gratis y legal

5.

PRUEBA EN DISTINTOS NAVEGADORES

Nunca va a quedar igual

6.

PACIENCIA... SI VA MAL ES FALLO TUYO :(

Estamos empezando

SI ALGO FALLA

1. Revisa los conceptos básicos de maquetación
2. Revisa la sintaxis de tu web.
3. Revisa los estilos por defecto.
4. Vuelve al uno hasta que todo esté bien.

Es decir **TEN PACIENCIA**

EL FLUJO DE LOS ELEMENTOS

HELLO!

I am Juan Diego Pérez

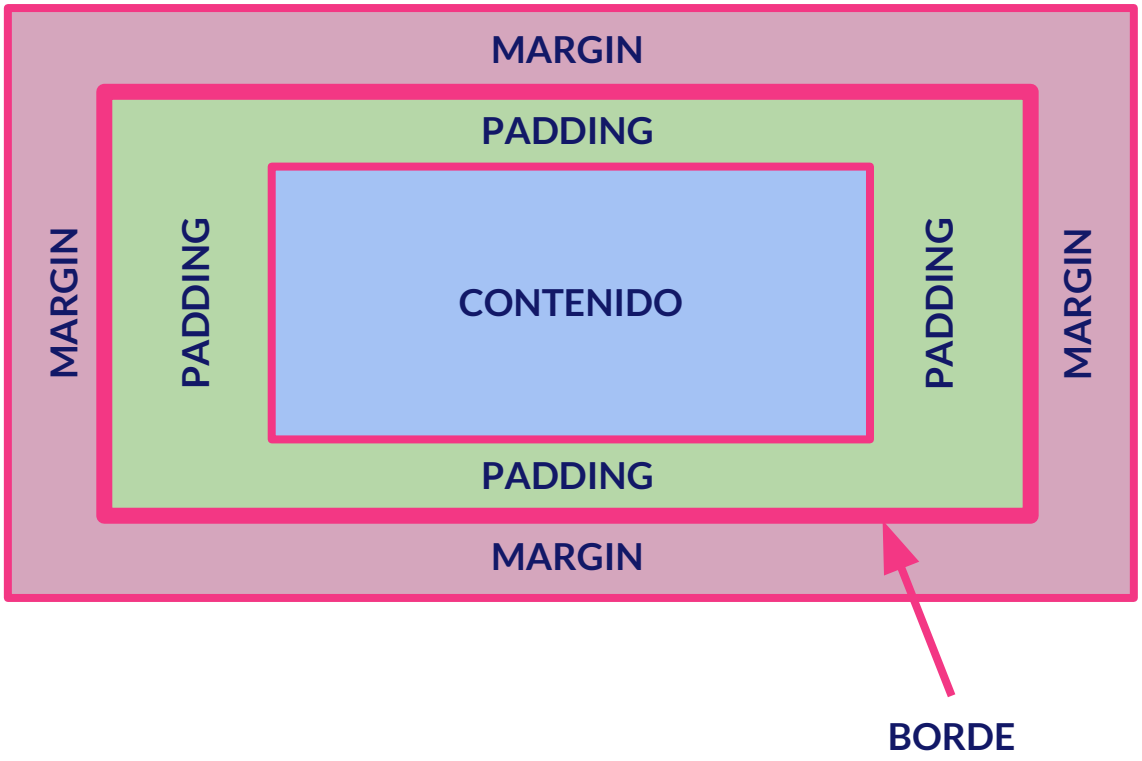
You can find me at @pekechis &

http://github.com/pekechis

1.

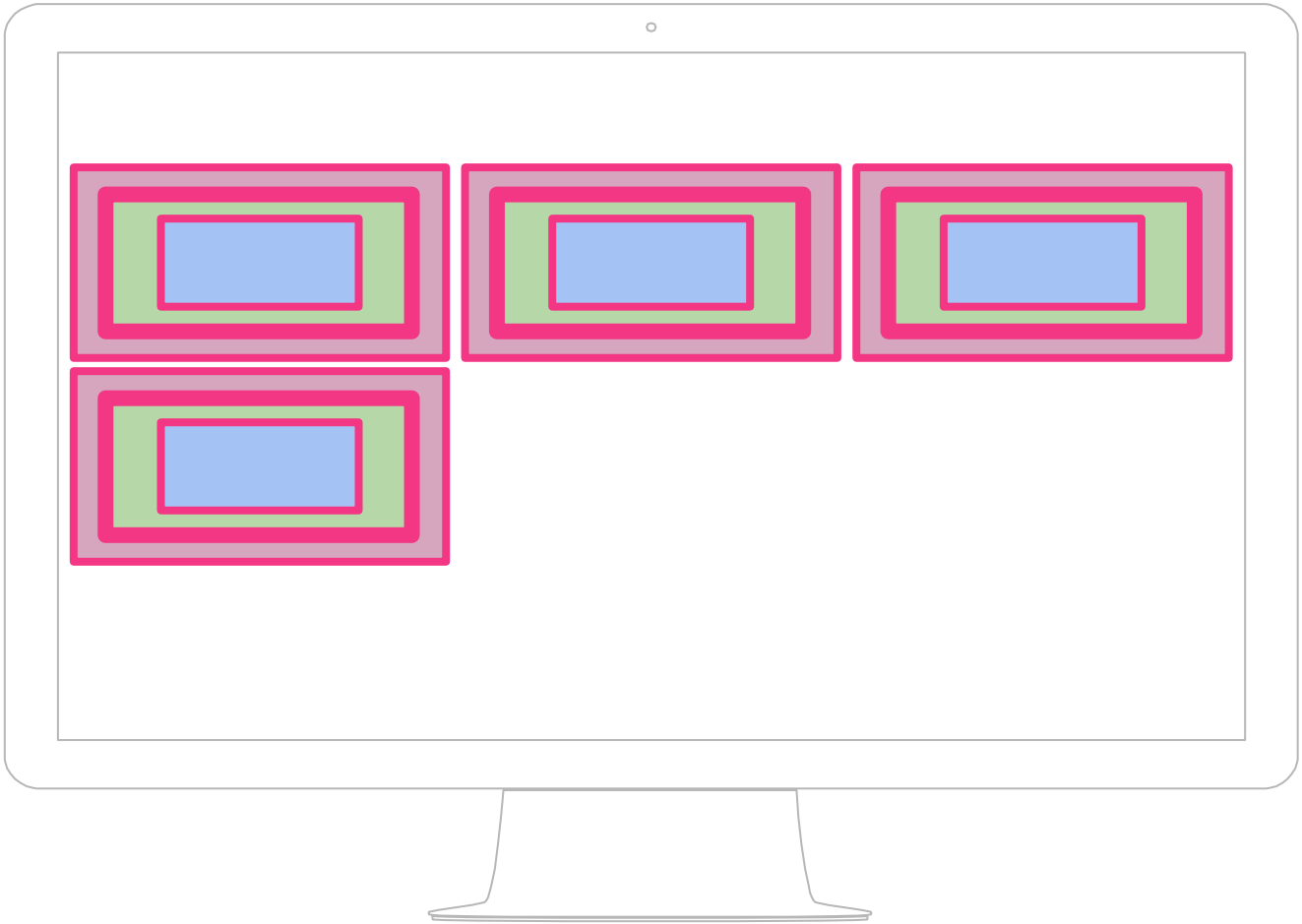
CONSIDERACIONES IMPORTANTES

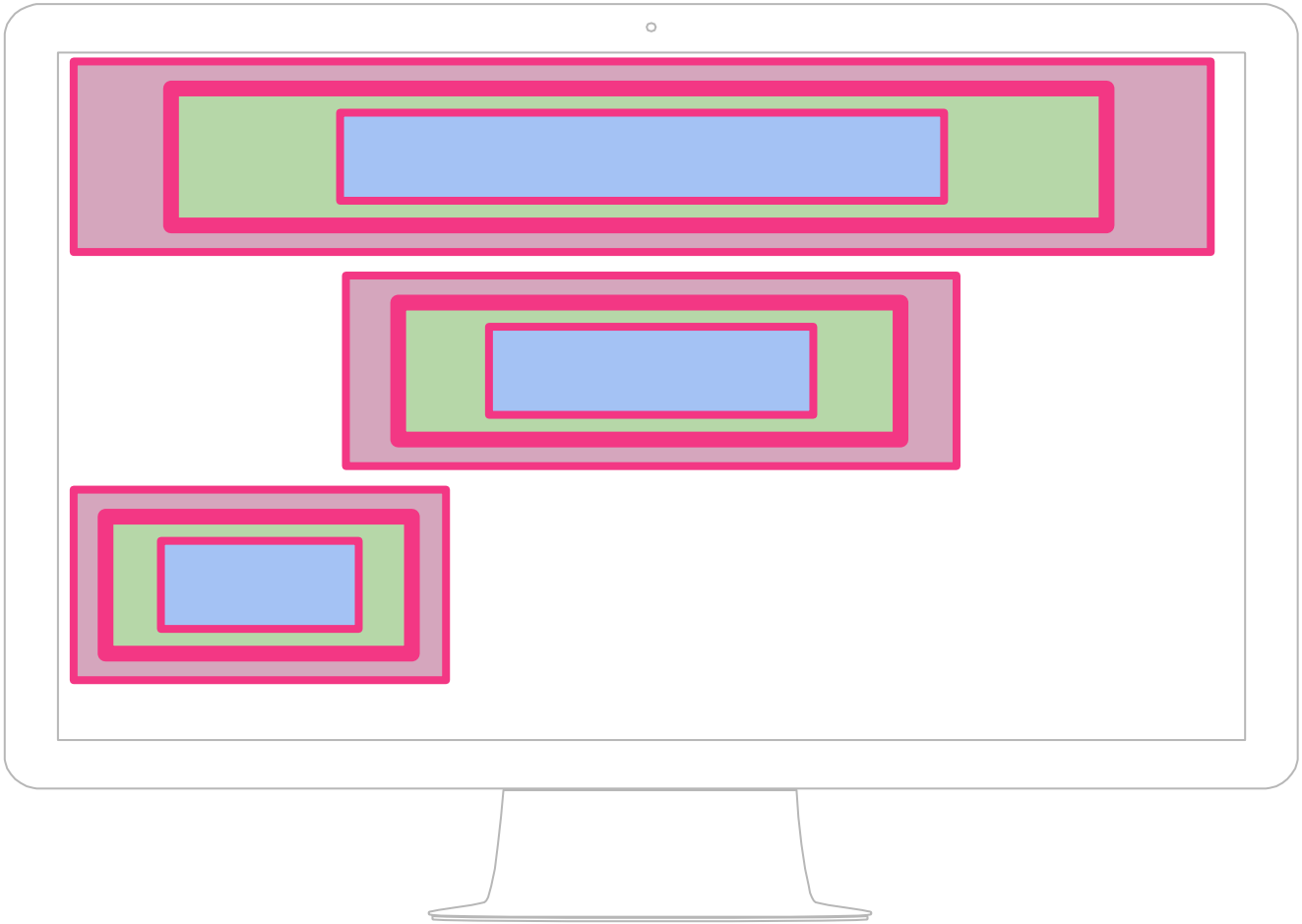
Debe quedar muy claro


MARGIN
PADDING
CONTENIDO
PADDING
MARGIN
PADDING
MARGIN
MARGIN
BORDE

ASPECTOS A RECORDAR

- Los navegadores no hacen **NADA** para controlar el diseño de nuestra página web.
- Sólo muestran los elementos en **ORDEN**.
- Siguen sólo dos reglas básicas...





2.

¿ENTONCES?

¿Cómo lo hago?

ENTONCES...

DISEÑO = CSS

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

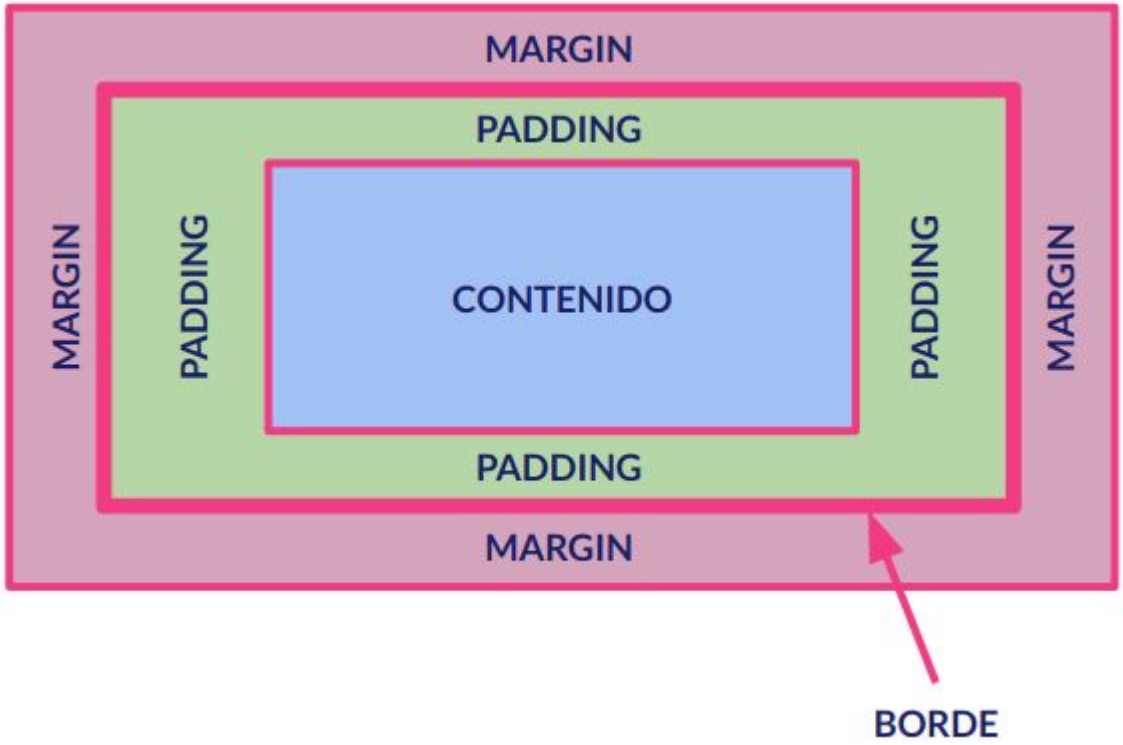
EN LÍNEA Y EN BLOQUE

PROPIEDAD DISPLAY

1.

TODO ES UNA CAJA

Recordamos


MARGIN
PADDING
MARGIN
PADDING
CONTENIDO
PADDING
MARGIN
PADDING
MARGIN
BORDE

¿CÓMO SE COMPORTAN?

Aunque todos los elementos son cajas no todas se comportan igual. Su comportamiento viene determinado por la propiedad display

VALORES DE DISPLAY MÁS USADOS

- inline
- inline-block
- block
- none
- Relacionados con tablas
- Flex y Grid (próximos cursos)

Y alguno más:

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

2.

INLINE, INLINE-BLOCK

Siguiendo la línea

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>...

3.

BLOCK

"Saltos de línea"

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>...

4.

NONE

Desapareciendo

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.

5.

RELATIVOS A TABLAS

Emulando el comportamiento de las tablas

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

TIPOS DE LAYOUTS

1.

TIPOS

¿Cuántos hay?

TIPOS DE LAYOUTS

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

2.

FIXED

Tamaño en pixels

“

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.

3.

ELASTIC

Tamaño en em

“

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

4.

FLUID

Tamaño en %

“

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 :(

5.

MAX/MIN WIDTH (HÍBRIDO)

Límites superior
e inferior

“

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

6.

RESPONSIVE

Cambia con la pantalla

“

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)

7.

¿CUÁL?

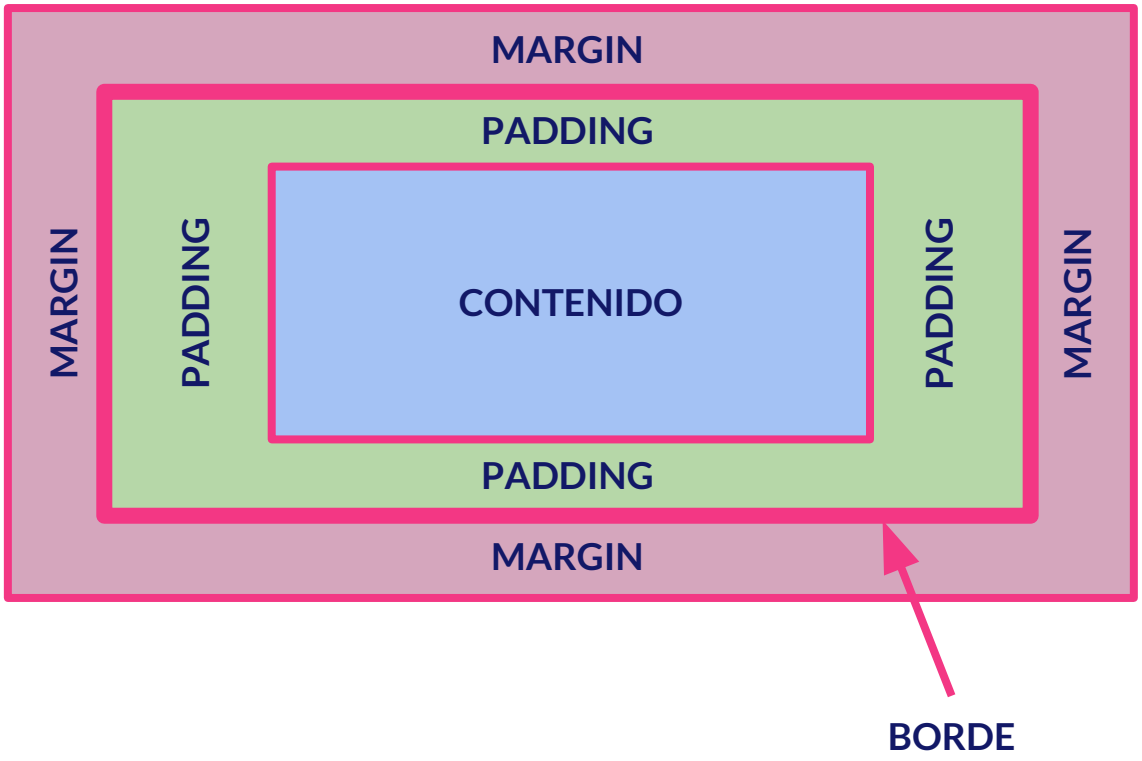
???????

LA PROPIEDAD
BOX-SIZING

1.

EL MODELO DE CAJA

Recordando...


MARGIN
PADDING
CONTENIDO
PADDING
MARGIN
PADDING
MARGIN
MARGIN
BORDE

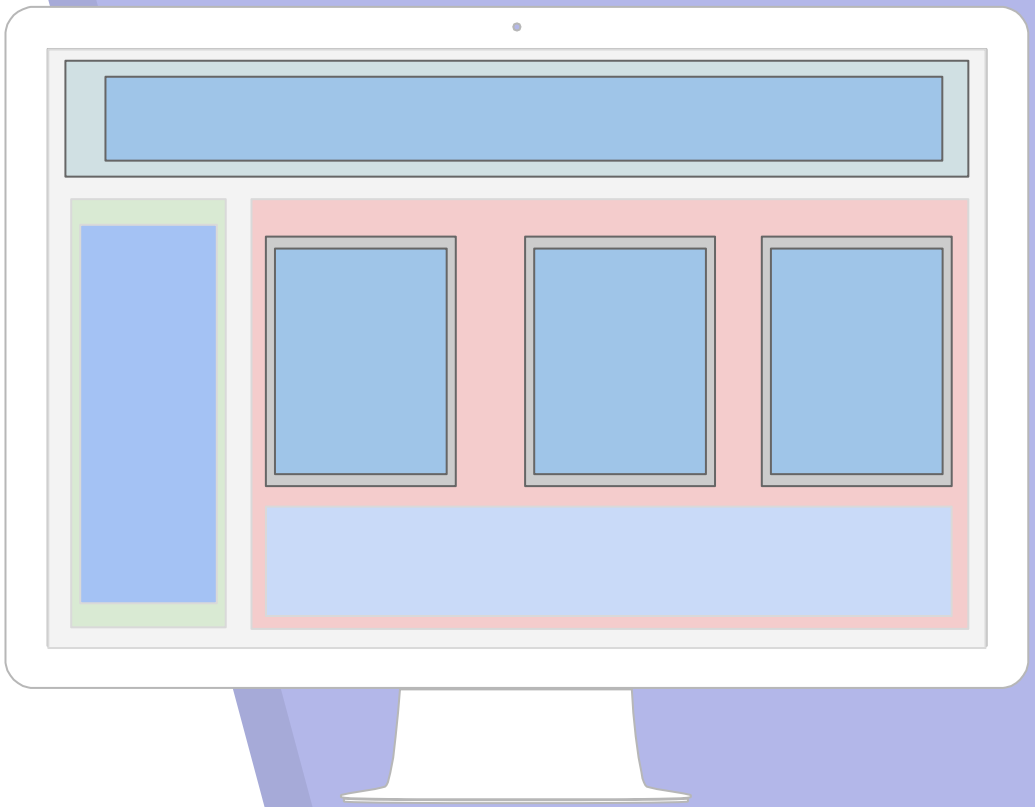
2.

MAQUETACIÓN

Recordando...

MAQUETACIÓN WEB

Disponer los elementos de nuestra web, su estructura



3.

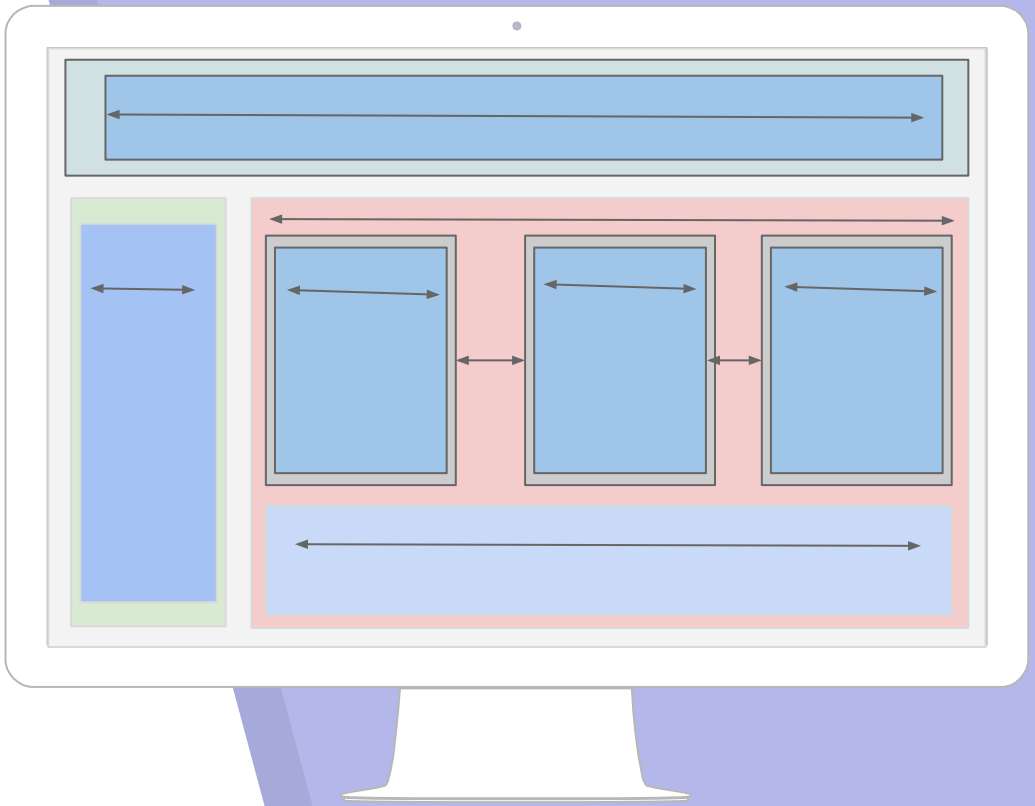
DIMENSIONES POR DEFECTO

Así calculan los navegadores

“

Altura del elemento = altura del contenido+ padding + borde

Anchura del elemento = anchura del contenido+ padding + borde

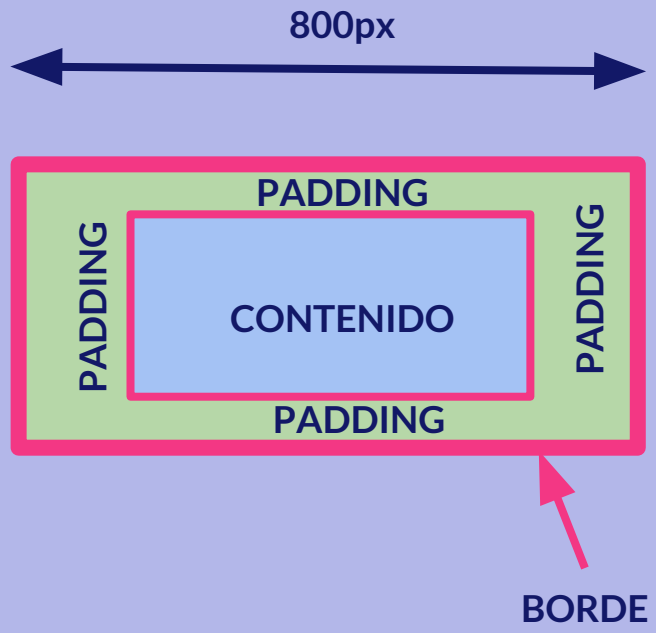


```
* {
 -webkit-box-sizing: border-box;
 -moz-box-sizing: border-box;
 box-sizing: border-box;
 }
```

UN VALOR
BORDER-BOX

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

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



4.

OTRAS POSIBILIDADES

Por si te aburres

OTRAS POSIBILIDADES DE BOX-SIZING

content-box

Es la forma por defecto, para saber que ocupa la caja debo sumar todo

padding-box

Debo sumarle únicamente el tamaño del borde para saber las dimensiones (aún no tiene soporte en navegadores)

5.

UN EJEMPLO

Por si no me crees

CENTRAR UN ELEMENTO

1.

CENTRAR UN ELEMENTO

Quiero que todo quede bonito

“

Cuando hablamos de centrar un elementos 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.

2.

CENTRADO HORIZONTAL

Quiero que todo quede bonito

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 (próximos cursos)**

3.

CENTRADO VERTICAL

Quiero que todo quede bonito

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 (próximos cursos)**

CENTRADO VERTICAL ELEMENTOS EN BLOQUE

- Utilizando la propiedad **position en el contenedor y en el elemento** (próximo vídeo)
- Con contenedores **flex (próximos cursos)**

4.

EJEMPLOS

Quiero que todo
quede bonito

PROPIEDAD POSITION. COLOCANDO ELEMENTOS

1.

POSICIONANDO ELEMENTOS

Para detalles
más finos

CON LA PROPIEDAD POSITION

- static
- relative
- absolute
- fixed
- sticky
- inherit

Y va unido a las propiedades top,bottom,left,right (desplazamiento) y z-index (capas)

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

2.

Z-INDEX

Capas y más capas

“

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

3.

CENTRADO VERTICAL CON POSITION (para elementos de bloque)

Ahora só

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%);
}`
```

4.

EJEMPLOS

En directo se
entiende mejor

PERO 

CUIDADO

Si posiciono elementos tengo que dar altura al contenedor de alguna manera

PERO 

CUIDADO

Algunos navegadores no dan soporte a todas las propiedades (sobre todo a sticky)

EJERCICIO
PRÁCTICO

Home Products About Us Contact

Home
Products
About Us
Contact

ELEMENTOS CON COLUMNAS

1.

DEFINICIÓN

¿qué es eso?

“

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

1.

PROPIEDADES CSS

¿qué necesito?

PROPIEDADES PARA COLUMNAS

- **column-count**: nº de columnas
- **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)

PROPIEDADES PARA COLUMNAS

- **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.

PERO 

CUIDADO

Algunos navegadores no dan soporte a todas las propiedades

ELEMENTOS FLOTANTES

1.

PROPIEDAD FLOAT

Otra forma de
crear estructuras

“

...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?

“

...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

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?

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?

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?

2.

PROPIEDAD CLEAR

Otra forma de crear estructuras

“

...si quiere ***“forzar”*** que un elemento deje de flotar debo añadir la la propiedad ***clear:right*** **(o left o both)** y ese elemento ya se añadirá tras el fín en vertical del elemento flotante...

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?

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?

3.

CLEARFIX HACK

También nos puede pasar posicionando...

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*/
}
```

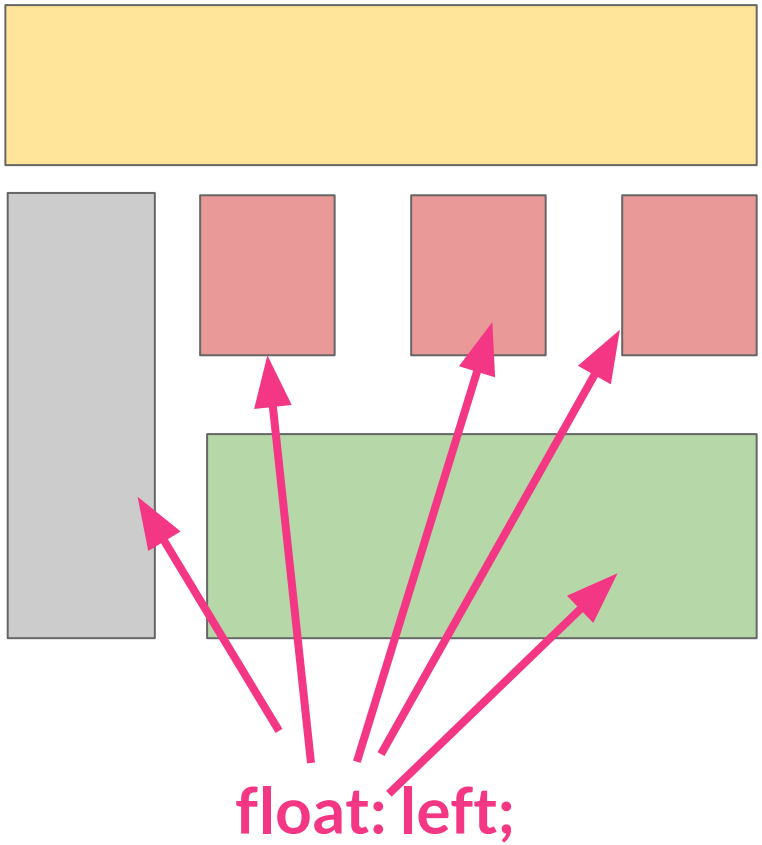
4.

CREANDO LAYOUTS

Otra forma de
crear estructuras

> 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.


float: left;

5.

EJEMPLO

EJERCICIO PRACTICO ELEMENTOS FLOTANTES


EJERCICIO PRÁCTICO ELEMENTOS FLOTANTES
Inicio
Productos
Sobre Nosotros
Contacta
Inicio
Productos
Sobre Nosotros
Contacta
CSS 3
Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime animi at, unde sit qui quam deserunt exercitationem voluptates nisi, ea harum ex necessitatibus voluptatum, vel commodi iusto repudiandae adipisci ratione?
CSS 3
Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime animi at, unde sit qui quam deserunt exercitationem voluptates nisi, ea harum ex necessitatibus voluptatum, vel commodi iusto repudiandae adipisci ratione?
CSS 3
Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime animi at, unde sit qui quam deserunt exercitationem voluptates nisi, ea harum ex necessitatibus voluptatum, vel commodi iusto repudiandae adipisci ratione?
Lorem ipsum dolor sit amet consectetur, adipisicing elit. Illum neque eligendi suscipit voluptates rerum quod. Voluptatem eaque sed magni? Vel quis alias suscipit, sed quo dolores. Animi numquam ut ad.
Lorem ipsum dolor sit amet consectetur, adipisicing elit. Illum neque eligendi suscipit voluptates rerum quod. Voluptatem eaque sed magni? Vel quis alias suscipit, sed quo dolores. Animi numquam ut ad.
Lorem ipsum dolor sit amet consectetur, adipisicing elit. Illum neque eligendi suscipit voluptates rerum quod. Voluptatem eaque sed magni? Vel quis alias suscipit, sed quo dolores. Animi numquam ut ad.
Curso desarrollado por @pekechis para OpenWebinars