HTML + CSS

Digital House >
Coding School

4

Episodio IV

Una "nueva" estructura para maquetar sitios web

Contenido

- Etiquetas HTML Semánticas
  - div, span
  - article, section, header, footer
- CSS - Modelo de caja
  - display: inline vs block
  - float
  - width / height
  - margin / border / padding
  - Box-sizing
- Dev Tools

Secciones semánticas

1. <section> </section>

Sección de contenido monotemático.

2. <article> </article>

Porción de información en una sección.

No semánticas

**1. **

División de contenido.

**2. **

Tag multifunción (no es un bloque).

Secciones semánticas

1. <header> </header>

Cabecera de contenido o documento.

2. <footer> </footer>

Pié de contenido o de documento.

No semánticas

**1. **

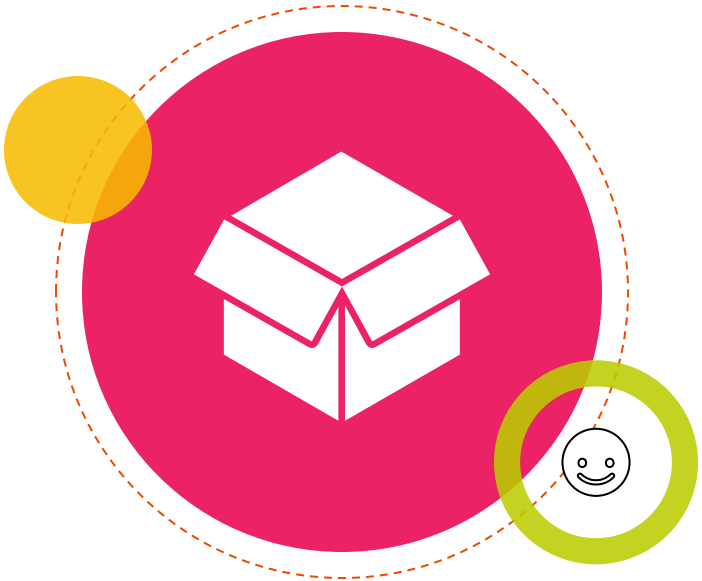
División de contenido.

**2. **

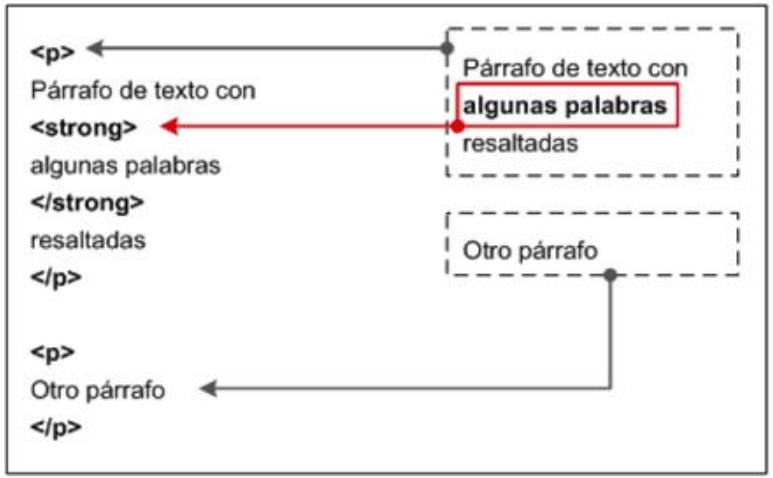
Tag multifunción (no es un bloque).

Realizar el punto #1 de la guía de ejercicios.

Modelo de caja



Seguramente la **característica más importante de CSS**



Modelo de caja

- Es el comportamiento que hace que todos los elementos de un documento HTML se representen mediante cajas rectangulares.
- Condiciona el diseño de todas las páginas web.

Tipos de Elementos

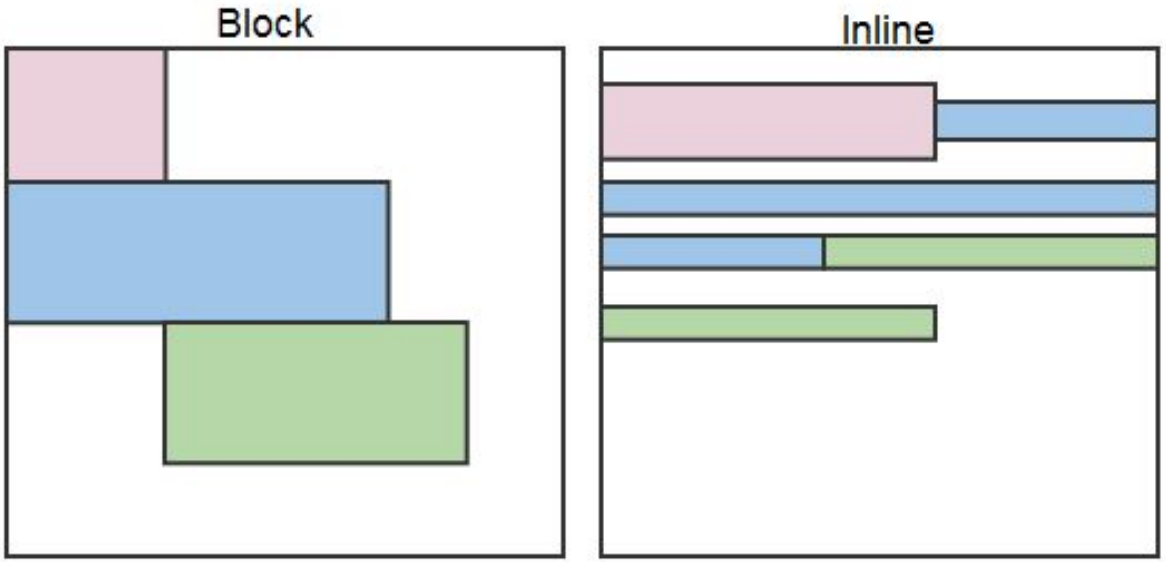
display: block, inline

inline

Define un elemento con comportamiento línea, no recibe algunas propiedades del modelo de caja.

block

Define un elemento con comportamiento de bloque, recibe “facilmente” propiedades del modelo de caja.



Podemos usar las **DEV TOOLS** de nuestro browser para entender mejor esto.

¿Qué son las DEV TOOLS?

Tipos de Elementos

display: inline-block, none

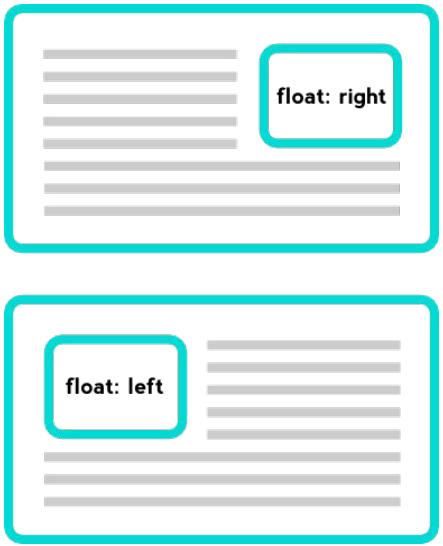
inline-block

Define un elemento con comportamiento de semi-bloque, recibe “fácilmente” propiedades del modelo de caja. Comparte también propiedades de elemento de línea.

none

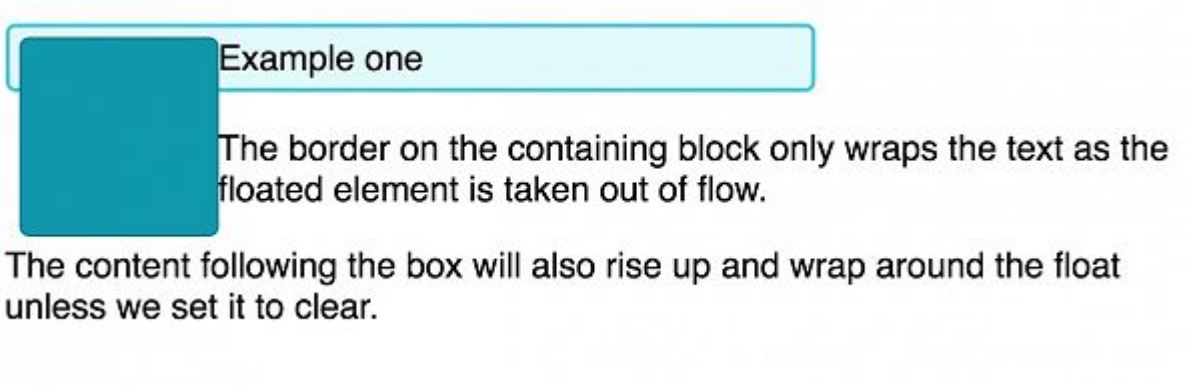
Oculta a un elemento en la visual. No lo elimina de la estructura de HTML.
Solo efecto visual.

Posicionamiento por flotación

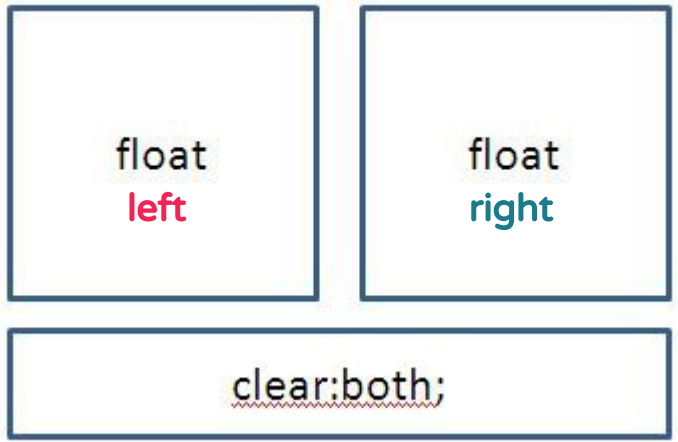


- Permite que los elementos de bloque "compartan la fila" dentro de la que se encuentran con otros elemento que también floten.
- Los elementos por debajo del elemento flotado, asumen que éste último no existe y tienden a ocupar el lugar vacío que el elemento flotante "ha dejado". NO sucede lo mismo si los elementos son de texto.

Posicionamiento por flotación



Posicionamiento por flotación



No flota, tiene clear: both;

Clear: es la forma más sencilla de "limpiar" flotaciones. Basta con aplicarle dicha propiedad al elemento que se encuentra por debajo de las cajas flotantes, para que éste conserve su posición.

Overflow

Controlando como “rebalsa”
el contenido de una caja.

Overflow

La propiedad overflow, nos permite controlar el “flujo” interno de un contenedor.

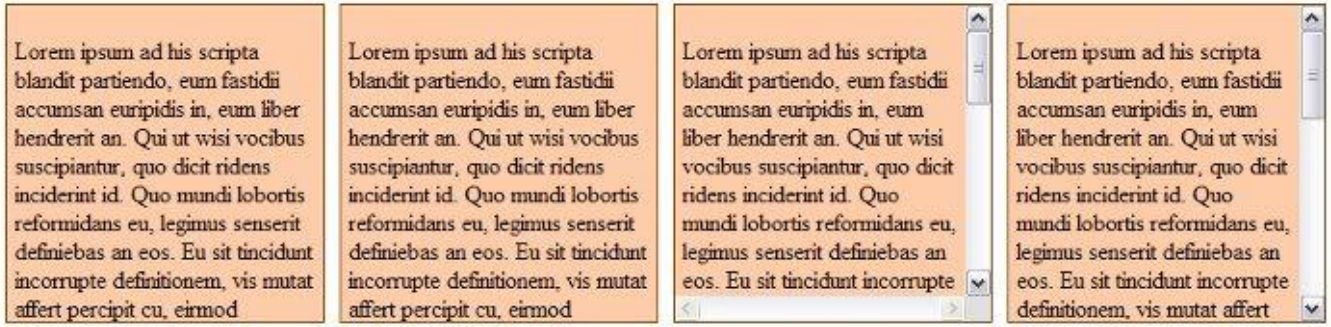
En otras palabras, nos permite manipular lo que sucede cuando el contenido interno rebosa en tamaño a su contenedor padre.

La propiedad overflow tiene 4 valores posibles:

- visible
- hidden
- scroll
- auto

overflow: visible;

overflow: scroll;



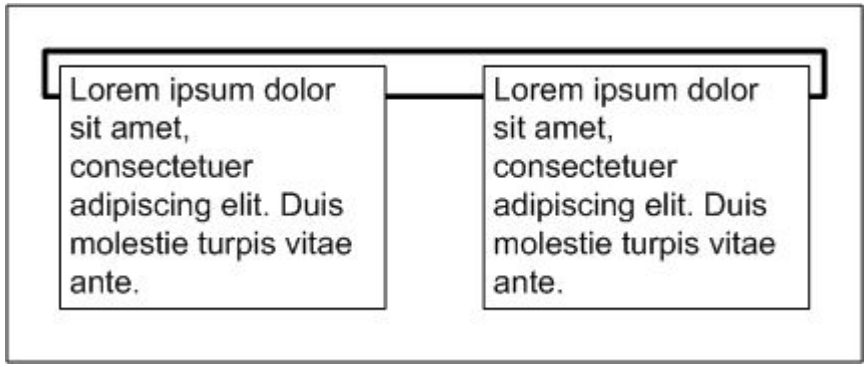
consectetuer signiferumque eu per. In usu latine equidem dolores. Quo no falli viris intellegam, ut fugit veritus placerat per.

Ius id vidit volumus mandamus, vide veritus democritum te nec, ei eos debet libris consulatu. No mei ferri graeco dicunt, ad cum veri accommodare. Sed at malis omnesque delicata, usu et iusto zzril meliore. Dicunt maiorum eloquentiam cum cu, sit summo dolor essent te. Ne quodsi nusquam legendos has, ea dicit voluptua eloquentiam pro, ad sit quas qualisque. Eos vocibus deserunt quaestio ei.

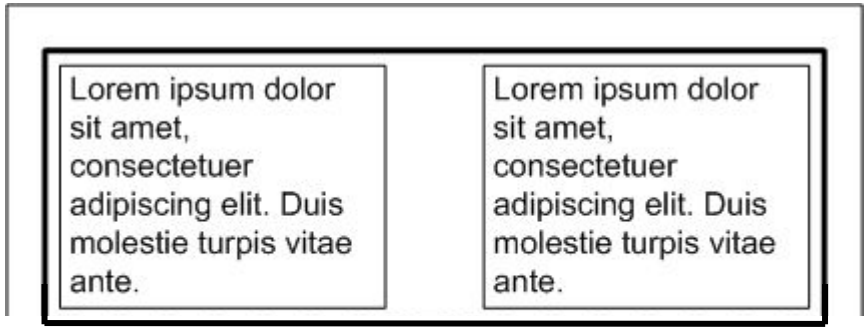
overflow: hidden;

overflow: auto;

Overflow para limpiar la flotación



Comportamiento natural del contenedor padre e hijos con flotación.



Comportamiento del contenedor padre con la propiedad overflow: hidden;

Propiedades CSS
== Width ==

```
elemento {

    width: 560px;


    /* podemos usar % */

    /* width: 50% */

}
```

Atenti: si un elemento no tiene declarado el witdh, el mismo será igual al 100% de su padre contenedor siempre y cuando sea un bloque.

Propiedades CSS
== Height ==

```
elemento {

    height: 200px;


    /* NO podemos usar % */

    /* height: 20% */

}
```

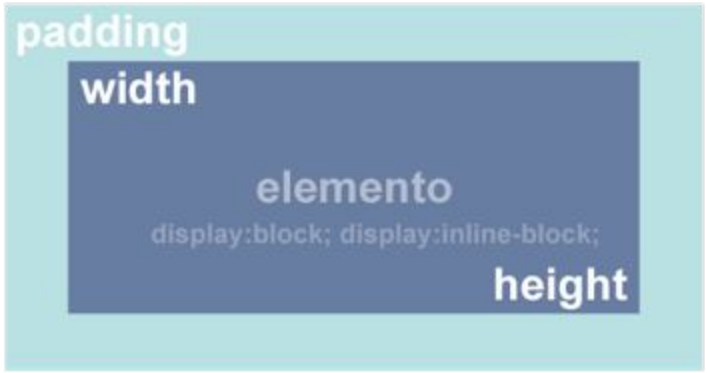
Atenti: si un elemento no tiene declarado el height, el mismo será igual a la altura que le proporcione su contenido interno. Sea un bloque o línea.


width
elemento
display:block; display:inline-block;
height

Propiedades CSS

== Padding ==

```
elemento {

    padding-top: 10px;
    padding-right: 20px;
    padding-bottom: 30px;
    padding-left: 40px;


    /* shorthand */

    padding: 10px 20px 30px 40px;

}
```


padding
width
elemento
display:block; display:inline-block;
height

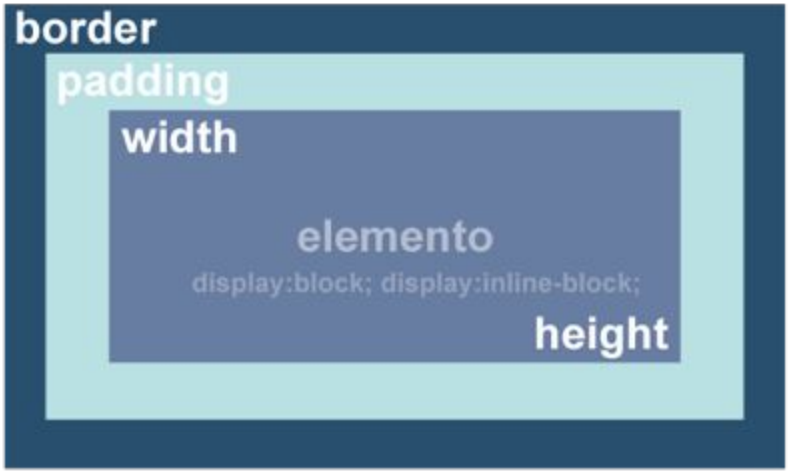
Propiedades CSS
== Border ==

```
elemento {

    border: solid 5px red;
}
```

Sintaxis

border: **style width color**;

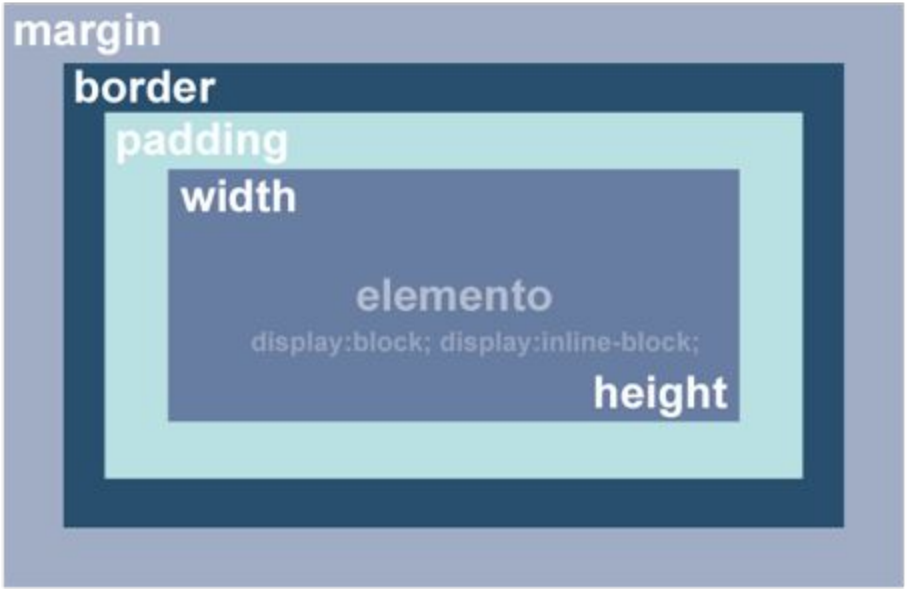
style => solid | dotted | dashed | double

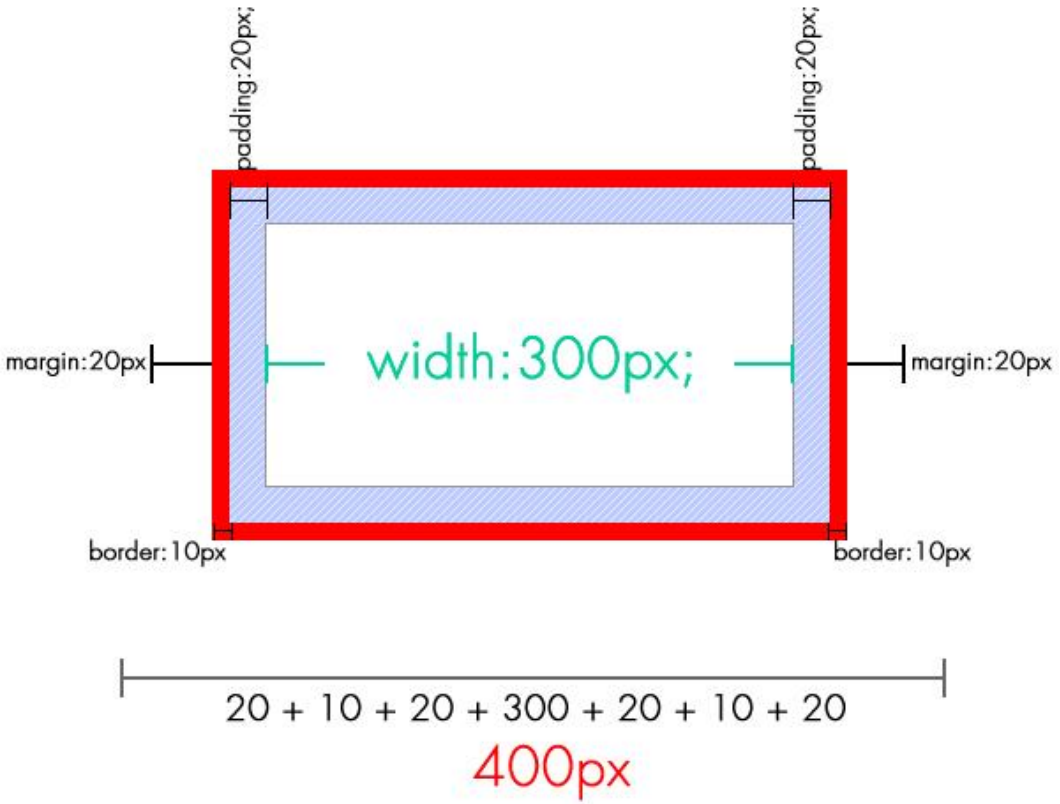

border
padding
width
elemento
display:block; display:inline-block;
height

Propiedades CSS

== Margin ==

```
elemento {

    margin-top: 10px;
    margin-right: 20px;
    margin-bottom: 30px;
    margin-left: 40px;


    /* shorthand */

    margin: 10px 20px 30px 40px;

}
```

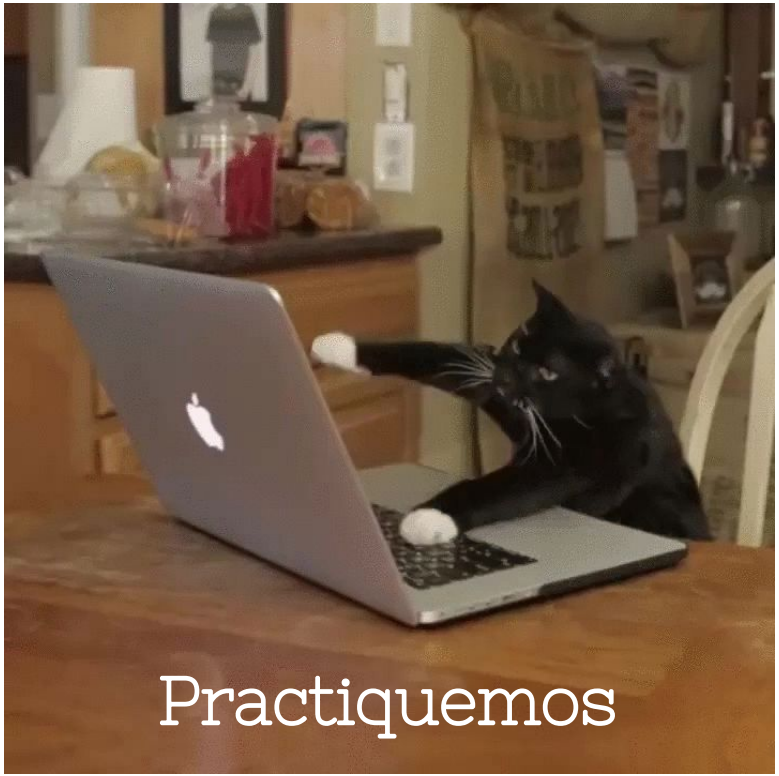

margin
border
padding
width
elemento
display:block; display:inline-block;
height


padding:20px;
padding:20px;
margin:20px
width:300px;
margin:20px
border:10px
border:10px
20 + 10 + 20 + 300 + 20 + 10 + 20
400px

Propiedades CSS
== Box-sizing ==

```
elemento {

    box-sizing: border-box;

    /*content-box*/

}
```

Definición: Esta propiedad permite que el **box-model** sea más fácil de usar, pues descuenta del ancho y alto automáticamente, lo que sumamos en relleno y borde. **Atenti: El margin se sigue sumando.**



Realizar desde el punto #2 hacia adelante de la guía de ejercicios.

Gracias!

¿Consultas?

Recuerden: javier@digitalhouse.com || @jap_solo