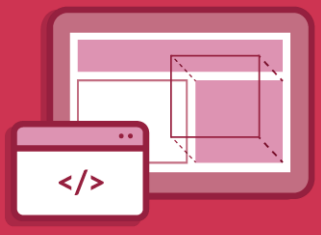
Diseño WEB

Clase 5

Box Modeling I



coderhouse

Modelo de cajas

Repaso anterior

- ¿Cuál es la función principal del CSS?
- ¿Qué es un elemento de línea?
- Entonces, ¿qué es un elemento de bloque?
- ¿Qué grupo de propiedades son heredadas de un elemento a sus hijos?
- ¿Cuál es la única etiqueta que no hereda las propiedades de texto?

Antes de escribir el HTML de un sitio, se debería **planificar la estructura**. Para esta planificación se utiliza el llamado **Wireframe** (esquematiza el diseño de página u ordenamiento del contenido del sitio web). No es otra cosa que un «boceto» sin diseño de cómo se quiere estructurar la web. **Facilita el proceso de detectar los elementos que necesitaremos y sus propiedades básicas**.
Dos excelente herramientas de Wireframe son http://balsamiq.com y http://creately.com/

El Maquetado Web es **convertir ese Wireframe a su versión clickeable (funcional)**.

Propiedades de caja

Todos los elementos del HTML son cajas. Un ****, un **<h2>** y demás, son rectangulares:

- En los elementos de línea, se verá uno al lado del otro.
- En los elementos de bloque, uno debajo del otro.

Ese concepto de que "todo es una caja" da lugar a algo llamado en web como box model. Sin importar si son de línea o de bloque, todas las etiquetas tienen las siguientes propiedades en común:

Estas 5 propiedades son numéricas, por lo cual serán un número y su respectiva unidad (px, em, cm…)

WIDTH: Espacio declarado para el ancho del contenido de la caja.

HEIGHT: Espacio declarado para el alto del contenido de la caja.

BORDER: Tipo de línea que envolverá la caja.

PADDING: Separación entre el borde y el contenido de la caja.

MARGIN: Separación entre el borde y el afuera de la caja.

Entonces de adentro hacia fuera, las cajas tienen contenido, padding, borde y margen.

- **Contenido (content)**: Es el contenido HTML del elemento, como los párrafos, las listas, las imágenes, etc.
- **Relleno (padding)**: Es el espacio libre (opcional) que existe entre el contenido y el borde.
- **Borde (border)**: Es la línea que encierra el contenido y su relleno.
- **Margen (margin)**: Es el espacio libre (opcional) entre la caja y el resto de las cajas adyacentes.

El espacio final que ocupa un elemento (ancho o alto) será la sumatoria de todos esos valores.

ESPACIO PARA EL CONTENIDO: Lo que digan el WIDTH / HEIGHT. **ANCHO FINAL DEL ELEMENTO:** padding-left + border-left + width + border-right + padding-right. **ALTO FINAL DEL ELEMENTO:** padding-top + border-top + height + border-bottom + padding-bottom.	**ANCHO y ALTO DEL ELEMENTO:** Lo que digan el WIDTH / HEIGHT. **ANCHO PARA EL CONTENIDO:** width - padding-left - border-left - border-right - padding-right. **ALTO PARA EL CONTENIDO:** height - padding-top - border-top - border-bottom - padding-bottom.

Alto y ancho

Por default, si no le indicamos ningún ancho específico, el navegador va a hacer que la caja ocupe todo el ancho de la ventana (*en los elementos de bloque*). Con respecto al alto, va a hacer que la caja ocupe el mínimo posible, por lo que el alto de la misma va a depender pura y exclusivamente del contenido que tenga.

Ancho

La propiedad CSS que controla la anchura de la caja de los elementos se denomina *width*.

La propiedad *width* no admite valores negativos y los valores en porcentaje se calculan a partir de la anchura de su elemento padre. El valor *inherit* indica que la anchura del elemento se hereda de su elemento padre.

El siguiente ejemplo establece como valor de ancho *200px* para el elemento **`<div>`** caja:

```
<div id="caja">
  ...
</div>
```

```
#caja { width: 200px; }
```

Otras dos propiedades relacionadas con la anchura de los elementos: *min-width* y *max-width*.

La propiedad *max-width* **define el ancho de una caja, sin darle un valor fijo**. De esta manera le podemos dar un tamaño de ancho máximo, **lo que significa que si la ventana del navegador se achica o se agranda, la caja también lo hace, pero con un ancho máximo de tope**. O sea, no se va a agrandar más que el valor que le hayamos configurado en la propiedad *max-width*.

```
<div id="caja">
  ...
</div>
```

```
#caja {
  width: 200px;
  max-width: 800px;
}
```

De la misma manera que definimos un ancho máximo, **podemos definir un ancho mínimo**. Lo que significa que la caja no va a medir menos de lo que nosotros le hayamos configurado. Esta propiedad se llama *min-width*.

```
<div id="caja">
  ...
</div>
```

```
#caja {
  width: 200px;
  max-width: 800px;
  min-width: 400px;
}
```

Para resumir, basándonos en el ejemplo anterior, nuestra caja no va a medir menos de *400px* ni más de *800px*, dentro de esas medidas y dependiendo del tamaño de la ventana del navegador, nuestra caja es variable.

Altura

Al igual que sucede con *width*, la propiedad *height* no admite valores negativos. Si se indica un porcentaje, se toma como referencia la altura del elemento padre. Si el elemento padre no tiene una altura definida explícitamente, se asigna el valor *auto* a la altura.

El siguiente ejemplo establece como valor de alto *50px* para el elemento ** ** cabecera:

```
<div id="cabecera">
  ...
</div>
```

```
#cabecera {
  height: 50px;
}
```

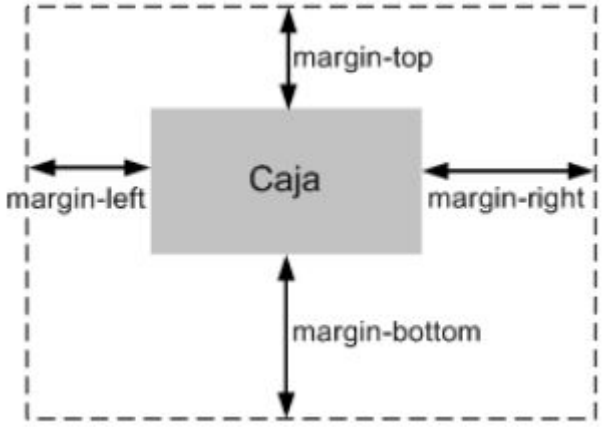
Otras dos propiedades relacionadas con la altura de los elementos: *min-height* y *max-height*.

Estas dos propiedad funcionan exáctamente igual que con el ancho. **Nos permite definir topes máximos y mínimos en la altura de nuestra caja.**

Márgenes

CSS define cuatro propiedades para **controlar cada uno de los márgenes horizontales y verticales de un elemento**.

Cada una de las propiedades establece la separación entre el borde lateral de la caja y el resto de cajas adyacentes:



Normalmente, las unidades más utilizadas para indicar los márgenes de un elemento son los píxeles.

Las propiedades *margin-top*, *margin-right*, *margin-bottom*, *margin-left* se utilizan para definir el ancho de los márgenes de cada uno de los lados del elemento por separado.

Puedes definir los 4 lados o solo aquellos que necesites.

En base a nuestro ejemplo anterior, vamos a definir márgenes a nuestro div caja.

```css
#caja {
  width: 200px;
  max-width: 800px;
  min-width: 400px;
  margin-top: 5px;
  margin-bottom: 5px;
  margin-left: 10px;
  margin-right: 10px;
}
```

Además de las cuatro propiedades que controlan cada uno de los márgenes del elemento, CSS define una propiedad especial que permite establecer los cuatro márgenes de forma simultánea. Esta propiedad se llama *margin*. Entonces:

Si sólo se indica un valor, todos los márgenes tienen ese mismo valor:

```css
#caja {
  margin: 5px;
}
```

En este caso, nuestra caja va a tener un margen de *5px* en sus cuatro lados.

Si se indican dos valores, el primero se asigna al margen superior e inferior y el segundo se asigna a los márgenes izquierdo y derecho:

```css
#caja {
  margin: 5px 10px;
}
```

En este caso, nuestra caja va a tener un margen de 5px de arriba y de abajo y 10 px de la derecha y de la izquierda.

Si se indican los cuatro valores, el orden de asignación es: margen superior, margen derecho, margen inferior y margen izquierdo:

```css
#caja {
  margin: 5px 10px 12px 15px;
}
```

En este caso, nuestra caja va a tener un margen de *5px* de arriba, *10px* de la derecha, *12px* de abajo y *15px* de la izquierda.

Relleno

CSS define cuatro **propiedades para controlar cada uno de los espacios de relleno horizontales y verticales de un elemento**.
Cada una de estas propiedades establece la separación entre el contenido y los bordes laterales de la caja del elemento.

Al igual que con los márgenes, CSS también define una propiedad llamada padding para establecer los cuatro rellenos de un elemento de forma simultánea.

Entonces, podemos escribir cada relleno por separado, definiendo su ubicación:

```
#caja {
  padding-top: 5px;
  padding-bottom: 5px;
  padding-left: 10px;
  padding-right: 10px;
}
```

O podemos escribir los cuatro juntos con la propiedad padding por sí sola:

```
#caja {
  padding: 5px 10px 12px 15px;
}
```

Posicionamiento y visualización

Cuando los navegadores descargan el contenido HTML y CSS de las páginas web crean una caja para representar a cada elemento. Los factores que se tienen en cuenta para generar cada caja son:

- Las propiedades *width* y *height* de la caja (si están establecidas).
- El tipo de cada elemento HTML (elemento de bloque o elemento en línea).
- Posicionamiento de la caja (normal, relativo, absoluto, fijo o flotante).

- Las relaciones entre elementos (dónde se encuentra cada elemento, elementos descendientes, etc.)
- Otro tipo de información, como por ejemplo el tamaño de las imágenes y el tamaño de la ventana del navegador.

Tipos de elementos

El estándar HTML clasifica a todos sus elementos en dos grandes grupos: elementos en línea (*inline*) y elementos de bloque (*block*).

Los elementos de bloque siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea. Por otra parte, los elementos en línea no empiezan necesariamente en nueva línea y sólo ocupan el espacio necesario para mostrar sus contenidos.

Los párrafos son elementos de bloque.

Los enlaces son elementos de línea.
(Dentro de un párrafo los enlaces siguen siendo elementos de línea.)

Los elementos en línea definidos por HTML son:
`a abbr acronym b basefont bdo big br cite code dfn em font i img input kbd label q s samp select small span strike strong sub sup textarea tt u var`

Los elementos de bloque definidos por HTML son:
`address blockquote center dir div dl fieldset form h1 h2 h3 h4 h5 h6 hr isindex menu noframes noscript ol p pre table ul`

Los siguientes elementos también se considera que son de bloque:
`dd dt frameset li tbody td tfoot th thead tr`

Alto explícito

En web, el alto fijo en los elementos no existe. Cuando un elemento tiene un alto fijo, cualquier contenido que exceda la caja será visible. Esto genera como problema que si después viene otro contenido, se van a superponer. Sólo debe declararse un *height* **si estas seguro que el contenido a mostrar no supera ese alto**.

Si de todas formas declarás un alto fijo, **podés definir qué hacer con el sobresaliente de la caja**. CSS tiene la propiedad *overflow* que maneja todo el contenido que sobresalga del *border* (visible o no).
Tiene 4 valores posibles:

- *visible*: Valor por defecto. El excedente es visible.
- *hidden*: El excedente no se muestra (lo corta).
- *scroll*: Genera una barra de scroll en los dos ejes (x/y) del elemento aunque no se necesite.
- *auto*: Genera el scroll solo en el eje necesario.

```
#caja {
  overflow: visible;
}
```

Hacer columnas

Los elementos de bloque se muestran por defecto uno por debajo del otro. Por más que le reduzcas el width, su comportamiento natural no es mostrarse uno al lado del otro. Si tu layout planea mostrar algún contenido en columnas, necesitas usar otras propiedades CSS. Básicamente hay dos técnicas:

- Convertir el elemento de bloque en uno de línea.
- Correr el objeto hacia un lado (derecha/izquierda).

Display

Se encarga de **definir cómo se ve un elemento HTML**. Los dos comportamientos más importantes son:

- Pasar un elemento de bloque a elemento de línea.
- Pasar un elemento de línea a elemento de bloque.

Eso se hace con los valores *block* e *inline* respectivamente:

- *block*: Convierte el elemento en elemento de bloque

- *Inline*: Convierte el elemento en elemento de línea

```
#caja {
  display: block;
}
```

```
#caja {
  display: inline;
}
```

Display Inline

Los elementos de línea se muestran uno al lado del otro pero tienen un problema: **No entienden los márgenes ni *padding* de arriba y abajo**. Si bien agrandan el elemento, éste se superpone a lo que esté arriba y abajo del mismo. Para que entienda el *padding* y margen de arriba y abajo, se usa *display: inline-block*. Se trata de un elemento de línea (uno al lado del otro) respetando todas las reglas físicas de uno de bloque.

```
#caja {
  display: inline-block;
}
```

Bug del display: inline-block

El display *inline-block* con ancho fijo pone las cosas una al lado de la otra (sí, lo dijimos recién). Pero si tenés los anchos milimétricamente calculados puede ser que el último aparezca abajo (y no al lado). En los elementos de línea (inline o inline-block) si entre los elementos hay espacio (sea uno o 500 enter, espacios o tab) **ese aire entre elementos se muestra como un espacio de barra espaciadora**. De ahí que no te entren y se muestre uno debajo de los demás.

Quitar un elemento

El display tiene también un valor para quitar un elemento del layout *display: none;* Lo oculta y, además, lo quita (no ocupa su lugar).

```
#caja {
  display: none;
}
```

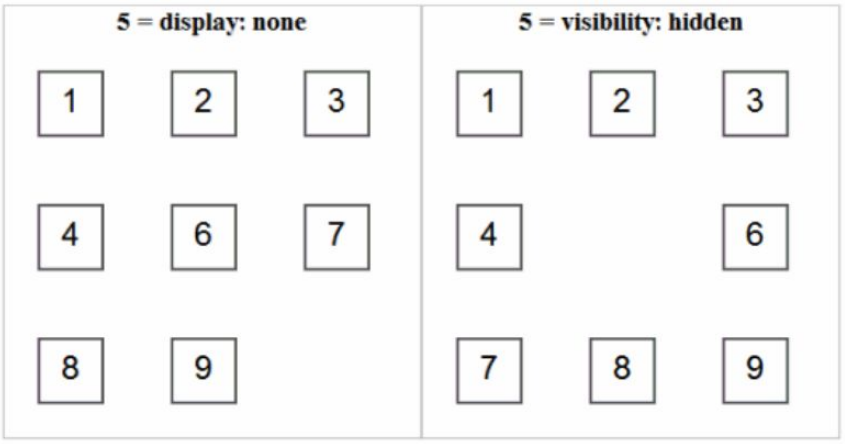
Y CSS tiene, además una propiedad llamada *visibility* que indica si un elemento se visualiza o no. Los valores posibles son

- *hidden* (oculto)
- *visible* (por defecto).

Un elemento con *visibility: hidden;* sigue ocupando su lugar en el layout aunque no se vea.

```
#caja {
  visibility: hidden;
}
```

```
#caja {
  visibility: visible;
}
```



Flotaciones

Float

La flotación es **mover un elemento hacia la derecha o izquierda de su línea y todo lo que viene después se acomodará en el "hueco" que queda vacío**. Es una manera 'old fashion' de hacer una columna. Se usa la propiedad *float* que acepta dos valores:

- *left*: Corre la caja a la izquierda.
- *right*: Corre la caja a la derecha.

IMPORTANTE: Cuando un elemento flota, deja de pertenecer al flujo normal del HTML.

```
#caja {
  float: left;
}
```

```
#caja {
  float: right;
}
```

El **`<div>` verde** tiene un ancho fijo y flota a la izquierda. Como los divs **rojo** y el **azul** son de bloque, se deben ver uno abajo del otro

Todo elemento flotado, deja de “empujar” en alto a su contenedor. Si todos los elementos flotan, el contenedor colapsa su altura.

Cómo solucionarlo.
Si googlean sobre el tema, van a encontrar muchas técnicas para solucionar el problema del float.
Al elemento que se colapsa, darle un *overflow* (excedente) con cualquier valor -menos *scroll*-

Clear

La propiedad *clear* permite **modificar el comportamiento por defecto del posicionamiento flotante** para forzar a un elemento a mostrarse debajo de cualquier caja flotante. La regla CSS que se aplica al segundo párrafo del ejemplo anterior es la siguiente:

```
#caja {
  clear: left;
}
```

La propiedad *clear* indica **el lado del elemento HTML que no debe ser adyacente a ninguna caja posicionada de forma flotante**. Si se indica el valor *left*, el elemento se desplaza de forma descendente hasta que pueda colocarse en una línea en la que no haya ninguna caja flotante en el lado izquierdo.

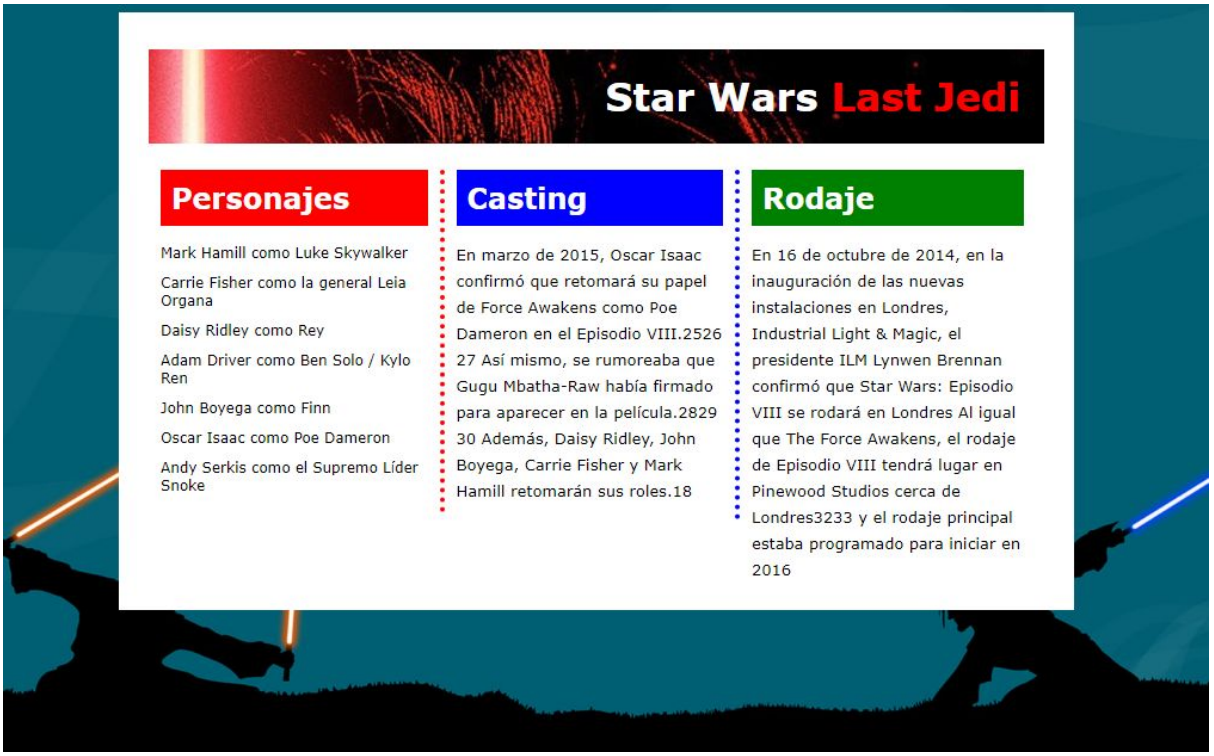
La especificación oficial de CSS explica este comportamiento como *"un desplazamiento descendente hasta que el borde superior del elemento esté por debajo del borde inferior de cualquier elemento flotante hacia la izquierda"*.

Si se indica el valor *right*, el comportamiento es análogo, salvo que en este caso se tienen en cuenta los elementos desplazados hacia la derecha.

El valor *both* **despeja los lados izquierdo y derecho del elemento, ya que desplaza el elemento de forma descendente hasta que el borde superior se encuentre por debajo del borde inferior de cualquier elemento flotante** hacia la izquierda o hacia la derecha.

Desafío

- Reproducir el siguiente Mockup, teniendo en cuenta la técnica de encolumnamiento de flotado:
  Imágenes:
  http://s000.tinyupload.com/index.php?file_id=56475481241895429028
  Textos:
  https://es.wikipedia.org/wiki/Star_Wars:_Episodio_VIII_-_Los_%C3%BAltimos_Jedi *(Duracion : 80 min)*



Desafío

- Agregar al CSS del proyecto (creado en el desafío 3) formato para color y tamaño de texto, espacio entre letras y espacio entre renglones.
- Generar columnas con los nuevos conceptos del Box Model aprendidos.
  Ejemplo: aplicar encolumnado a la presentación de los integrantes de la banda.