

E HTML PUBLIC "-//I

```
</d>  
<meta name="TITLE"  
<meta name="KEY  
<meta name="p  
<link rel="1  
<script  
</head>  
<body bo
```

FUNDAMENTOS DE CSS PROFESIONAL

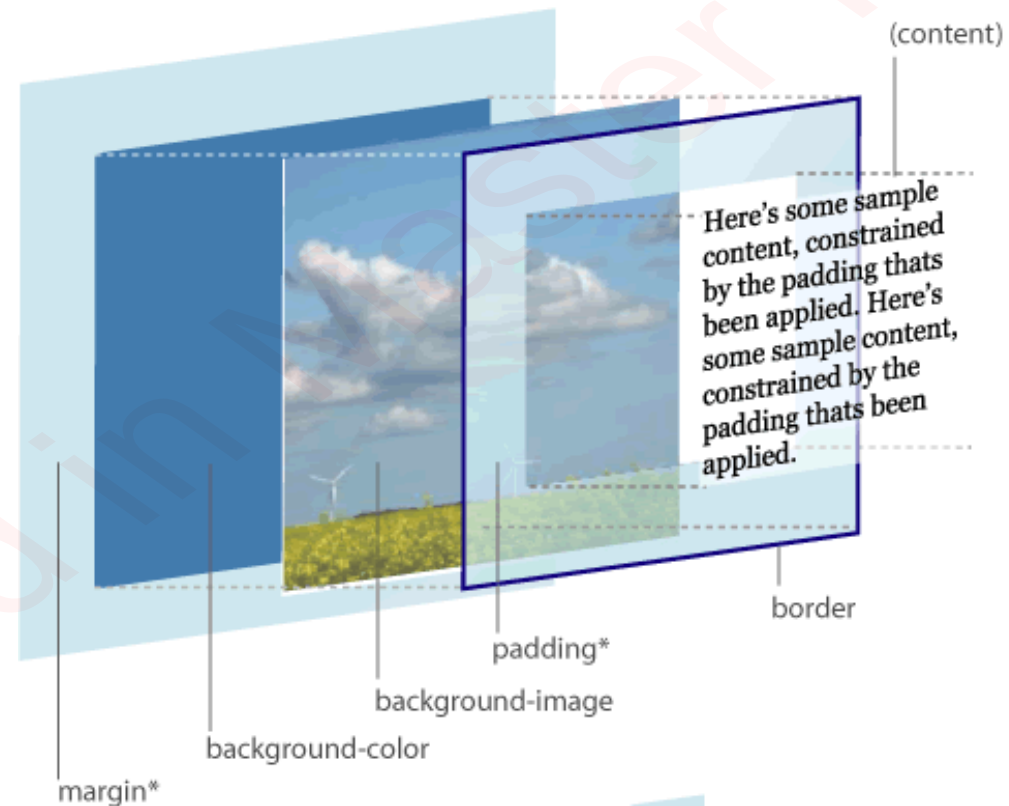
TEMA 1

DISPLAY

1.0. INTRODUCCIÓN

MODELO DE CAJA

THE CSS BOX MODEL HIERARCHY



1.0. INTRODUCCIÓN

MODELO DE CAJA

WIDTH | HEIGHT = WIDTH | HEIGHT OF CONTENT -> **MUERTE LENTA**

BOX-SIZING

box-sizing: content-box; (Muerte lenta)

box-sizing: padding-box; (Muerte regular)

box-sizing: border-box; -> **VIDA LARGA Y FELIZ**

WIDTH | HEIGHT = CONTENT + PADDING + BORDER

Ejemplo 1.1 - Box sizing

1.1. ELEMENTOS EN LÍNEA Y EN BLOQUE

Elementos en **LÍNEA** (display: inline)

- ▶ Ocupan el espacio necesario para mostrar su contenido.
- ▶ No tienen dimensiones modificables (alto, ancho).
- ▶ Permiten otros elementos a su lado.
- ▶ Padding y margin solo empujan a elementos adyacentes en horizontal, NUNCA EN VERTICAL.

1.1. ELEMENTOS EN LÍNEA Y EN BLOQUE

Elementos en **BLOQUE** (display: block)

- ▶ Ocupan todo el ancho disponible.
- ▶ Tienen dimensiones modificables (alto, ancho).
- ▶ No permiten otros elementos a su lado (aunque especifique un ancho, siguen ocupando todo el espacio disponible).

1.2. VALORES INTERESANTES DE DISPLAY

DISPLAY: INLINE-BLOCK

- ▶ Mezcla de inline y block (sorpresa!!).
- ▶ Tienen dimensiones modificables (alto, ancho).
- ▶ Permiten otros elementos a su lado. Si no especifico dimensiones, ocupa lo que ocupe su contenido.

1.2. VALORES INTERESANTES DE DISPLAY – DISPLAY: INLINE-BLOCK

ALGUNAS CONSIDERACIONES

- ▶ **VERTICAL ALIGN:** Alinea elementos ENTRE ELLOS (no su contenido dentro)
- ▶ **ESPACIOS “NO DESEABLES”** entre elementos y en imágenes dentro de contenedores.

Múltiples soluciones:

<https://davidwalsh.name/remove-whitespace-inline-block>

Ejemplo 1.2 - Display inline y block

Ejemplo 1.3 - Display inline-block, vertical align

Ejemplo 1.4.1 - Display inline-block: Espacios entre elementos

Ejemplo 1.4.2 - Display inline-block: Espacios bajo imágenes

1.2. VALORES INTERESANTES DE DISPLAY

DISPLAY: TABLE Y TABLE-CELL

- ▶ Me permite utilizar las propiedades del layout de tablas sin su rigidez -> cambio el comportamiento cambiando el display -> Puede ser **RESPONSIVE**
- ▶ Puedo mostrar datos de manera más semántica (header, sections, articles...)
- ▶ Lo vamos a usar principalmente para:
 - Menús justificados
 - Columnas de igual altura
 - Alineaciones verticales (En otro tema)

Ejemplo 1.5 - Menú

Ejemplo 1.5.1 MENÚ JUSTIFICADO PLANETA TRIATLÓN

Ejemplo 1.6 - Columnas igual altura

1.3. ANEXO: MULTICOLUMN LAYOUT

- ▶ Divide el contenido en columnas según le indique número (column-count) o ancho (column-width).
- ▶ OJO CON EL SOPORTE y los PREFIJOS. Verificar siempre todos los navegadores porque es guerrero.

SINTAXIS

`column-count: 3;` (Número de columnas)

`column-gap: 3rem;` (Separación entre columnas)

`column-rule: 2px solid black;` (Línea de separación)

`column-width: 150px;` (NO %. Es una SUGERENCIA y valor MÍNIMO, si no encaja no hace caso)

1.3. ANEXO: MULTICOLUMN LAYOUT

SINTAXIS

column-count: 3; (Número de columnas)

column-gap: 3rem; (Separación entre columnas)

column-rule: 2px solid black; (Línea de separación)

column-width: 150px; (NO %. Es una SUGERENCIA y valor MÍNIMO, si no encaja no hace caso)

Ejemplo 1.7 : MULTI-COLUMN LAYOUT - Texto en columnas

TEMA 2

POSICIONAMIENTO

2.0. INTRODUCCIÓN

- ▶ La propiedad **POSITION**, junto con la propiedad **FLOAT**, permite modificar la posición de cualquier elemento de la página.
- ▶ El navegador coloca cada elemento teniendo en cuenta el orden en el que aparece en el HTML y si es un elemento en línea o en bloque. Con **POSITION Y FLOAT** puedo modificar ese comportamiento.

2.1. POSITION

VALORES

- ▶ **STATIC:** Valor por defecto. Se ignoran top, bottom, left, right y z-index.
- ▶ **RELATIVE:** Posicionamiento de la caja respecto de su posición original.
 - El resto de elementos de la página respetan su posición original pero ignoran la nueva posición -> **POSIBLES SOLAPAMIENTOS.**
 - Top, right, bottom, left **desplazan la caja de su POSICIÓN INICIAL.**
 - Sirve de "referencia" para hijos posicionados de forma absoluta.

2.1. POSITION

VALORES

- ▶ **ABSOLUTE:** Posicionamiento de la caja respecto del 0,0 del primer ancestro que tenga establecida su posición diferente de STATIC.
- **FUERA DEL FLUJO NORMAL:** El resto de elementos **NO** respetan su posición original -> **SE MUEVEN** e ignoran la nueva posición -> **POSIBLES SOLAPAMIENTOS.**
- Top, right, bottom, left colocan la caja **respecto del primer ancestro con position establecida** -> si no hay -> **HTML**
- **OJO con el ancho** (igual en fixed), ocupa lo que ocupe el contenido -> si es una capa vacía -> me quedo sin ancho.

2.1. POSITION

VALORES

- ▶ **FIXED:** Posicionamiento de la caja respecto del 0,0 **DE LA VENTANA.**
 - **FUERA DEL FLUJO NORMAL:** El resto de elementos **NO** respetan su posición original -> **SE MUEVEN** e ignoran la nueva posición -> **POSIBLES SOLAPAMIENTOS.**
 - **Ni el BODY ni el HTML SON LA VENTANA,** por eso el elemento posicionado fixed no se menea aunque haga scroll.

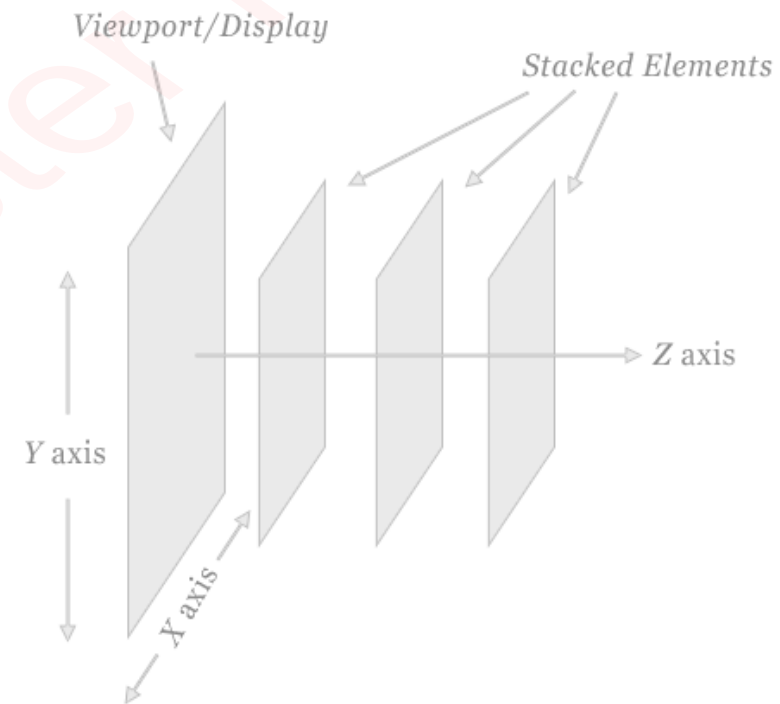
Ejemplo 2.1 - Position relative y absolute

Ejemplo 2.2 - Fixed

2.1. POSITION

Z-INDEX

- ▶ Determina el nivel de pila de un elemento HTML. El “nivel de pila” se refiere a la posición del elemento en el eje Z
- ▶ **SOLO EN ELEMENTOS CON POSITION**



Ejemplo 2.3: Z-INDEX

EJERCICIOS DE APLICACIÓN: VANITY FAIR: CAJAS “TORCIDAS”

2.2. FLOAT: POSICIONAMIENTO FLOTANTE

- ▶ **Se desplazan todo lo posible hacia la izquierda o hacia la derecha** de la línea en la que se encuentran.
- ▶ Si hay **otras cajas flotantes, NO SE SUPERPONEN ENTRE SÍ**, se colocan donde caben.
- ▶ **FUERA DEL FLUJO NORMAL:** pero de una manera especial y también hay "SOLAPAMIENTOS" pero de manera especial.

Otros elementos ocupan el lugar del elemento flotado:

- Elementos **EN LÍNEA:** **Hacen sitio** adaptando su ancho al espacio libre dejado por el elemento flotado
- Elementos **EN BLOQUE:** No les hacen sitio (ocupan su lugar) pero **ADAPTAN su contenido** para no solapar con el elemento flotado.

2.2. FLOAT: LIMPIAR FLOATS

- ▶ **CLEAR:** Los elementos que siguen a una caja flotada fluyen a su alrededor, CLEAR CONTROLA ESE COMPORTAMIENTO:
 - CLEAR: RIGHT -> NO HAGAS CASO A LAS CAJAS FLOTADAS A LA DERECHA.
 - CLEAR: LEFT -> NO HAGAS CASO A LAS CAJAS FLOTADAS A LA IZQUIERDA.
 - CLEAR: BOTH -> NO HAGAS CASO A NINGUNA CAJA FLOTADA.
- ▶ “Cuando un elemento tiene a todos sus hijos flotando, pierde su altura” -> **OVERFLOW, TÉCNICAS CLEARFIX.**

Ejemplo 2.4: FLOATS, CLEAR Y CLEARFIX

TEMA 3

CENTRAR

3.1. CENTRAR EN HORIZONTAL

1. DISPLAY INLINE-BLOCK + TEXT-ALIGN CENTER DEL PADRE
2. DISPLAY BLOCK + ANCHO CONOCIDO -> MARGIN LATERAL AUTO
3. ANCHO DESCONOCIDO: DISPLAY:TABLE + MARGIN LATERAL AUTO
4. ANCHO DESCONOCIDO: POSITION ABSOLUTE LEFT 50% + TRANSLATEX(-50%) (ojo con la altura del padre)

EJEMPLO 3.1: CENTRAR HORIZONTAL

3.2. CENTRAR EN VERTICAL

1. TEXTO (DE UNA SOLA LÍNEA) + ALTO DEL PADRE CONOCIDO -> LINE-HEIGHT = ALTURA DEL PADRE
2. SI PUEDO PONER AL PADRE DISPLAY: TABLE -> DISPLAY: TABLE-CELL + VERTICAL-ALIGN: MIDDLE
3. ALTO CONOCIDO -> POSITION: ABSOLUTE 50% + MARGIN TOP NEGATIVO 1/2 DEL ALTO (¡¡¡NO PORCENTAJES!!!)
4. ALTO DESCONOCIDO -> POSITION ABSOLUTE TOP 50% + TRANSFORM: translateY(-50%)

EJEMPLO 3.2: CENTRAR VERTICAL

3.3. CENTRAR EN HORIZONTAL Y VERTICAL

1. `DISPLAY TABLE-CELL + VERTICAL ALIGN MIDDLE + TEXT ALIGN CENTER`
2. `DIMENSIONES CONOCIDAS -> POSITION: ABSOLUTE 50% 50% + MARGIN NEGATIVO 1/2 DEL ALTO y 1/2 ANCHO(!!!NO PORCENTAJES!!!)`
3. `TODO DESCONOCIDO -> POSITION ABSOLUTE 50% 50% + TRANSLATE(-50%, -50%)`

EJEMPLO 3.3: Centrado horizontal y vertical

3.4. ENLACES DE INTERÉS

- ▶ <http://howtocenterincss.com>
- ▶ <https://css-tricks.com/centering-css-complete-guide/>
- ▶ <http://css-tricks.com/centering-in-the-unknown/>

EJERCICIO DE APLICACIÓN: Lightbox Oficina Wifi Santander

EJERCICIO DE APLICACIÓN: (VARIANTE POPUP) Lightbox Oficina Wifi Santander

(APARTADO 1.4 DISPLAY)

FLEXBOX

RESUMEN DE PROPIEDADES

Codepen interactivo con un **resumen de todas las propiedades** que afectan tanto al flex container como a los items:

Otros enlaces de interés:

- ▶ Flex-grow, flex-shrink, flex-base con sus fórmulas:
<https://chriswrightdesign.com/experiments/flexbox-adventures/>
- ▶ Solved by flexbox:
<https://philipwalton.github.io/solved-by-flexbox/>
- ▶ A complete guide to flexbox - CSS Tricks:
<http://css-tricks.com/snippets/css/a-guide-to-flexbox/>

EJERCICIO DE APLICACIÓN: FLEXBOX - Planeta Triatlón menú + sección "TRIATLÓN Y MÁS"

TEMA 4

PSEUDO-ELEMENTOS BEFORE Y AFTER

4.0. INTRODUCCIÓN

Pseudo-elementos ::before y ::after

- ▶ **Añado contenido** como primer hijo (BEFORE) o como último (AFTER) al elemento al que se lo aplico.
- ▶ Es **INDISPENSABLE** establecer la propiedad **CONTENT** para que funcione, aunque esté vacía:

content: ' '; LO PRIMERO

4.0. INTRODUCCIÓN

Diferencia pseudo-elementos y pseudo-clases

- ▶ **Pseudo-clase:** Se refiere a un **ESTADO TEMPORAL** de un elemento que **EXISTE en el DOM**. (Puedo indicar cuál es en mi HTML)
- ▶ **Pseudo-elemento:** **NO EXISTE EN EL DOM**. Es contenido que genero dinámicamente desde el CSS. (No puedo indicar cuál es en el HTML)

EJERCICIO DE APLICACIÓN: Links animados Guijuelo

EJERCICIO DE APLICACIÓN: Cuadrados responsive Planeta Triatlón + Autor con viñeta y subrayado especial

TEMA 5

ICONOS

5.1. ICON-FONTS

- ▶ Usar librerías ya hechas:
 - CDN
 - Descarga y aplicación con @font-face
- ▶ Crear mi propia librería:
 - **Icomoon** (la que yo utilizo)
 - Fontello
 - Fontastic

EJERCICIO DE APLICACIÓN (proyecto externo): Iconos_planeta_triatlon.zip

5.2.0. SVG INTRODUCCIÓN

▶ W3C DEFINICIÓN:

“SVG es un lenguaje para describir gráficos bidimensionales en XML”

Es decir, **un SVG es CÓDIGO.**

▶ SVG vs ICON-FONTS:

- <https://css-tricks.com/icon-fonts-vs-svg/>

5.2.1 SVG COMO CUALQUIER OTRA IMAGEN

- ▶ **Usar SVG dentro de ** -> Ventaja de la calidad y el peso pero mismas limitaciones que png o jpg

``

- ▶ **Usar SVG como background desde el CSS** -> Idem
`background: url(image.svg);`

No puedo modificar nada interno del SVG

5.2.2 SVG EN LÍNEA (EMBEBIDO)

1. **ESCALAR** con **ICOMOON**

2. **OPTIMIZAR**

- ▶ **APP:** <https://sourceforge.net/projects/svgcleaner/>
- ▶ **ONLINE:** <https://jakearchibald.github.io/svgomg/>

3. Insertar el código **SVG** en el **HTML**

Ejemplo 5.1: SVG embebido

5.2.2 SVG SPRITES EN LÍNEA

1. **ESCALAR** con **ICOMOON** y descargar zip
2. **demo.html** -> Incluir el **SVG (symbols)** en el **BODY**
3. Referenciar cada icono mediante el tag **<use>**

```
<svg class="svg-coaching">  
  <use xlink:href="#svg-coaching"></use>  
</svg>
```

Ejemplo 5.2: SVG SPRITES EN LÍNEA

5.2.3 SVG EXTERNO

1. **ESCALAR** con **ICOMOON** y descargar zip
2. **symbol-defs.svg** ES MI SVG EXTERNO
3. **demo-external-svg.html** - Referenciar cada icono mediante el tag **<use>** con RUTA e ID

```
<svg class="svg-coaching">
```

```
<use xlink:href="ruta/symbol-defs.svg#svg-coaching">
```

```
</svg>
```

Ejemplo 5.3: [svg_externo.zip](#)

TEMA 6

TRANSICIONES Y ANIMACIONES CSS

6.0. INTRODUCCIÓN

▶ CSS TRANSFORM

- MODIFICAN LA VISUALIZACIÓN DEL ELEMENTO CAMBIANDO SUS COORDENADAS Y DIMENSIONES
- **NO VEO EL PROCESO DE TRANSFORMACIÓN**
- 2D Y 3D (NOSOTROS 2D)

▶ CSS TRANSITION

- EL ELEMENTO PASA DE UN ESTADO A OTRO DONDE CAMBIAN SUS PROPIEDADES CSS. **SÍ VEO Y CONTROLLO EL PROCESO DE TRANSFORMACIÓN**, la **TRANSICIÓN** (ANIMACIÓN SENCILLA)

▶ CSS ANIMATION

- ANIMACIONES COMPLEJAS.

6.1. CSS TRANSFORM

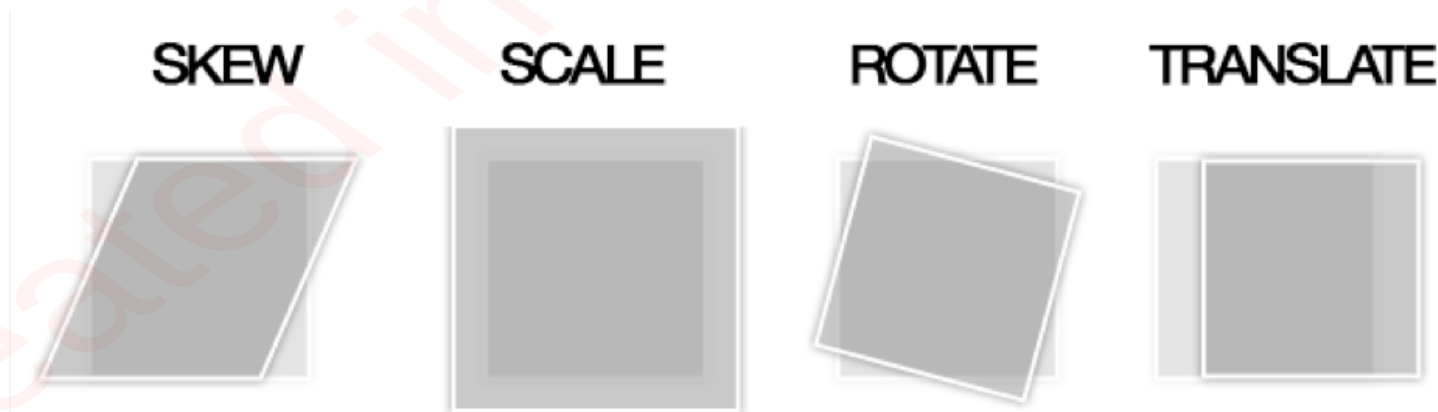
Dos propiedades:

- ▶ **TRANSFORM-ORIGIN**

transform-origin: bottom left / 10% 20% / 12px 3px;

- ▶ **TRANSFORM**

transform: scale(1.5);



6.2. CSS TRANSITION - SINTAXIS

transition: [propiedad/all] [duración] [timing-function] [delay]

Ej: transition: background-color .5s ease-out 1s;

► Shorthand de:

transition-property

transition-duration

transition-timing-function

transition-delay

6.2. CSS TRANSITION - SINTAXIS

- ▶ **PUEDO ANIMAR 1 PROPIEDAD:**

transition: background-color 500ms ease-out 1s;

- ▶ **VARIAS:**

transition: color .5s ease-out 1s, width 100ms ease;

- ▶ **TODAS:**

transition: all 500ms ease-out 1s;

- ▶ **INCLUSO "TRANSFORM", que también es una propiedad:**

transition: transform 500ms ease-out 1s;

- ▶ **PUEDEN ANIMARSE LAS PROPIEDADES QUE SEAN COLORES O NÚMEROS**

6.2. CSS TRANSITION - ¿CÓMO?

ESTADO INICIAL:

```
div {  
    transition: background-color 0.5s ease;  
    background-color: red;  
}
```

ESTADO FINAL:

```
div:hover ( o div.class) {  
    background-color: blue;  
}
```

Ejemplo 6.1: TRANSFORM Y TRANSITION

6.3. CSS ANIMATIONS - SINTAXIS

► SINTAXIS EN EL SELECTOR:

animation: [name] [duration] [timing-function] [delay]
[iteration-count] [direction] [play-state][fill-mode]

.selector {

**animation: nombreAnim 3s ease .5s infinite
alternate running forward;**

}

6.3. CSS ANIMATIONS – SINTAXIS

► SINTAXIS DE LA ANIMACIÓN (KEYFRAMES):

From -> To:

```
@keyframes nombreAnim {  
  from {  
    background-color: yellow;  
  }  
  to {  
    background-color: blue;  
  }  
}
```

Porcentajes de progreso:

```
@keyframes nombreAnim {  
  0% {  
    background-color: yellow;  
  }  
  50% {  
    background-color: blue;  
  }  
  100% {  
    background-color: red;  
  }  
}
```

6.3. CSS ANIMATIONS – PROPIEDADES

► Propiedades y sus valores:

- CSS TRICKS - ANIMATION

<https://css-tricks.com/almanac/properties/a/animation/>

- ENTENDER ANIMATION-FILL-MODE

<https://www.sitepoint.com/understanding-css-animation-fill-mode-property/>

Ejemplo 6.2: ANIMATION

EJERCICIO DE APLICACIÓN: PERSONAJES GUIJUELO

TEMA 7

RESPONSIVE

7.0. INTRODUCCIÓN

PUNTOS FUNDAMENTALES:

- ▶ **ETIQUETA VIEWPORT**

`<meta name="viewport" content="width=device-width, initial-scale=1">`

- ▶ **LAYOUT FLEXIBLE** : **Unidades relativas** para (casi) todo.

- ▶ **MEDIA QUERIES**

- ▶ **IMÁGENES**: max-width, svg, srcset, picture...

7.1. ETIQUETA VIEWPORT

“Ayuda” a los dispositivos a mostrar la web correctamente.

YES

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

NO

```
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
```


7.2. LAYOUT FLEXIBLE

- ▶ Anchos, margins y paddings en %, em, rem, vw o vh.

TARGET / CONTEXTO = RESULTADO EN %

- ▶ ¿Cuándo %?

- Cuando necesito que el valor sea un **porcentaje del alto o ancho del contenedor**.
- **OJO!!! padding y margin verticales** se refieren al **ANCHO** del contenedor.
- **OJO 2!!! height: 100%** (o cualquier otro porcentaje) **NO HACE NADA SI EL PADRE NO TIENE ALTO ESPECIFICADO O HEREDA (REGLA PRESENTE) DE SUS ANCESTROS**

Ejemplo 7.1: Porcentajes

7.2 LAYOUT FLEXIBLE: EM vs REM

REM: se refiere al tamaño de fuente establecido en el **HTML** y si no se ha establecido, al del **NAVEGADOR**.

- ▶ **¿Por qué REM y no EM?**

Cuando no quiera que un valor (**height, line-height, padding, margin...**) esté afectado por un cambio de **font-size** en el elemento o en sus ancestros. (Es un tamaño "fijo" hasta que cambie el font-size del HTML).

- ▶ **¿Por qué REM y no PX?**

Si un usuario incrementa o disminuye el tamaño de su fuente en el navegador, **las proporciones del layout se mantendrán**.

Ejemplo 7.2: REM

7.2 LAYOUT FLEXIBLE: EM vs REM

EM: se refiere al tamaño de fuente establecido **EN EL PROPIO SELECTOR** y está afectado por la **HERENCIA**.

► ¿Cuándo EM?

- Cuando quiera que un valor (height, line-height, padding, margin...) crezca o disminuya proporcionalmente con un cambio de font-size en el elemento o en sus ancestros.
- **En MEDIA QUERIES:** Siempre EN LAS MEDIA QUERIES se refiere al font-size del NAVEGADOR

Ejemplo 7.3: EM

7.3. MEDIA QUERIES

Establecemos condiciones a partir de las cuales se aplicarán reglas CSS distintas.

- ▶ **MOBILE FIRST:** Partimos del CSS que se aplicará en las resoluciones más pequeñas y modificamos con **MIN-WIDTH**

```
@media all/screen and (min-width: breakpoint){  
}
```

7.3. MEDIA QUERIES

► ¿DÓNDE?

- **Agrupadas por breakpoints:** más mantenible, menos líneas. Preferible para CSS estándar.
- **En contexto:** work-flow mucho más rápido, más flexible, con preprocesadores.

7.4. IMÁGENES

- ▶ **MAX-WIDTH (max-height):** Imágenes fluidas. Consigo que **no salga de su contenedor**, pero que tampoco exceda **SU PROPIO TAMAÑO** para **NO PIXELAR**.

```
img{  
  max-width: 100%;  
  height: auto;  
}
```
- ▶ **DISTINTOS TAMAÑOS:** srcset, etiqueta picture, media queries... (mucha información en internet, soporte aún limitado -> os lo dejo como deberes)
<https://css-tricks.com/responsive-images-youre-just-changing-resolutions-use-srcset/>
- ▶ **Usar SVG cuando podamos:** Ligeras y calidad óptima para todas las resoluciones

Ejemplo 7.4: Imágenes fluidas

EJERCICIO DE APLICACIÓN: [responsive_example.zip](#)