UT2: LM Y ESTILOS (2ª parte)

DIW – FRANCISCO CORBERA NAVAS

ÍNDICE

- 1. GUÍA DE ESTILO
- 2. LENGUAJE DE MARCAS HTML
- 3. HTML5
- **4. ESTILOS**

ESTILOS

- 1.- ¿Qué es CSS?
- 2.- Historia de CSS
- 3.- Insertar CSS en HTML
- 4.- Sintaxis y tipo de selectores
- 5.- Prepos – Prefijos Propietarios y Especificidad
- 6.- Herencia, Cascada - uso correcto
- 7.- Especificidad
- 8.- Resetear estilos de los browsers
- 9.- Metodología de escritura de CSS - BEM
- 10.- BOX-MODEL (Caja-Contenedor)

ESTILOS

- **1.- Qué es CSS**
  - Cascade Style Sheet – Hoja de estilo en cascada
  - CSS no es un lenguaje de programación, aunque tiene variables y funciones
    - No pueden crearse funciones nuevas aunque si usar las ya existentes
      - Ejemplo: calc
  - Único lenguaje de diseño gráfico que existe

ESTILOS

- **2.- Historia**
  - 1970 se crea un lenguaje para dar estilos a los documentos de SMGL (predecesor de HTML)
  - 1994/1995 – Se unen SSP y CHSS para definir CSS
    - Se presentaron propuestas para crear un lenguaje de estilos y estás 2 quedaron finalistas, uniéndose al final
  - 1996 – La W3C publica el estándar CSS1
    - Consorcio W3C regula como hay que escribir los estilos
  - 1998 – La W3C publica el estándar CSS2

ESTILOS

- **2.- Historia**

  - 2009 – La W3C publica el estándar CSS2.1
    - Sólo se añaden estilos nuevos
  - 2011 – CSS
    - La W3C empieza a lanzar SNAPSHOTS en vez de estándares →Mismo núcleo pero distintas ramas avanzando independientemente (Flexbox, Grid, Animaciones ….)

ESTILOS

2.- Historia

- CSS3 revoluciona el mundo web
- Hasta CSS3 no hacía falta hacer diseño Responsives porque no había móviles con internet.
- Al llegar los smartphones nace el Responsive Web Design gracias a CSS3
  - Flexbox, Grid

ESTILOS

- **3.- Insertar CSS en HTML**
  - **OPCIÓN 1: CSS interno**
    - En el head a través de la etiqueta
      - <style> código css </style>
    - Si tenemos una sola página con un estilo único si tiene sentido utilizarlo
    - Si tenemos varias páginas habría que hacer esto en cada una de ellas, por tanto NO ES LO MÁS RECOMENDABLE

ESTILOS

- **3.- Insertar CSS en HTML**
  - **OPCIÓN 2: CSS en línea**
    - En línea dentro de la etiqueta como atributo de HTML
      - <p style="código css">
    - Sólo es recomendable si el estilo de css tiene que cambiar en tiempo real, no para estilos estáticos
      - Se usa junto a Javascript para cambiar anchos, altos… cálculos en tiempo real en función de como esté la página

ESTILOS

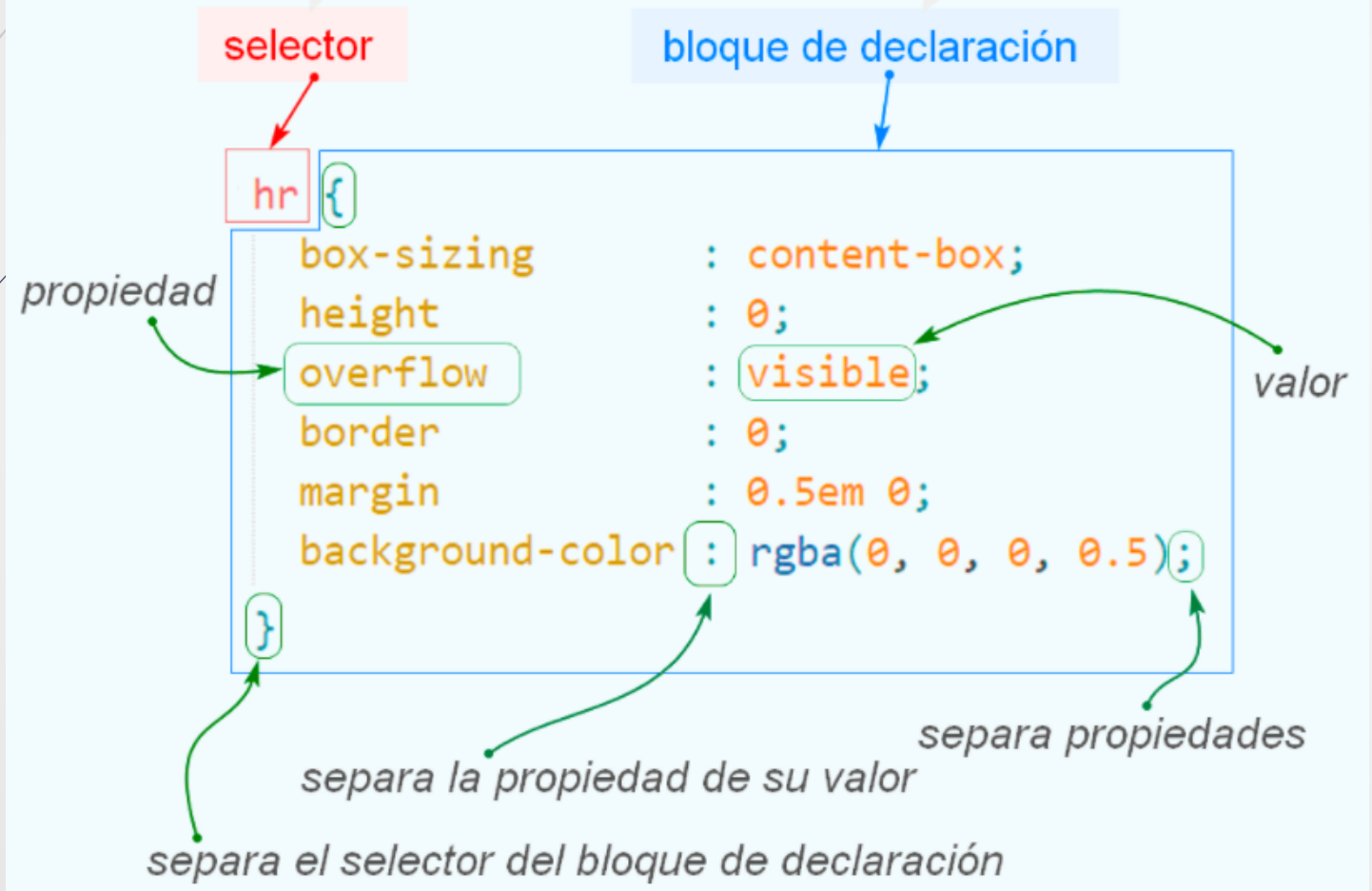
- **3.- Insertar CSS en HTML**
  - **OPCIÓN 3: CSS externo**
    - Asociando una hoja de estilos externa a nuestro documento
      - <link rel="stylesheet" href="styles.css" />
    - SIEMPRE se recomienda usar esta forma

ESTILOS

- **3.- Insertar CSS en HTML**
  - **OPCIÓN 4:**
    - A través de @import dentro de las etiquetas style del head
      - <style>
        - @import 'css/import.css'
      - </style>
    - **HANDICAP!** Si falla el import no se carga la web. Esperar a que se descargue el CSS para seguir obteniendo elementos
    - NO LO HAREMOS NUNCA pero está bien saberlo por si nos toca un proyecto en la empresa donde se haya usado

ESTILOS

4.- Sintaxis



ESTILOS

- **4.- Sintaxis**

  - No hay límite de propiedades que podemos añadir dentro de un selector

  - El **;** separa propiedades
    - Si solo hay una propiedad no es obligatorio
    - El último elemento tampoco necesita obligatoriamente el ;

  - Aunque no sea obligatorio es muy recomendable poner el ; al final para evitar errores y que no falle el CSS

ESTILOS

- **4.- Sintaxis**
  - At-rules
    - Reglas especiales que referencian a reglas css preprogramadas en el lenguaje.

      @media{}

      @font-face{} → importar fuentes de Google Fonts por ejemplo

      @keyframes{} → animaciones

      @import() → La que hemos visto antes, puede llevar paréntesis

    - Lista de propiedades

ESTILOS

- **4.- Sintaxis y tipo de selectores**
  - **Selectores de etiqueta**
    - **PROBLEMA** → Se aplican a todas las etiquetas
    - Si que se utilizan para resetear los estilos del navegador
    - No es recomendable usarlo para estilo internos más organizados

```
/*Selector de etiqueta*/
p{
    color: red;
}
h1{
    color: blue;
}
```

ESTILOS

- **4.- Sintaxis y tipo de selectores**
  - **Selectores de clase**
    - Se aplica a los elementos que utilizan el atributo class.
    - RECOMENDABLE usarlo siempre por especificidad (lo vemos más adelante lo que es)
    - Se referencian con un punto delante del nombre de la clase

```
/* Selector de clase */
.title{
    color: green;
}

.text{
    color: blueviolet;
}
```

ESTILOS

- **4.- Sintaxis y tipo de selectores**
  - **Selectores de id**
    - Igual que las clases pero se referencia con #
    - Para dar estilos mejor no utilizarlos nunca
    - Solo se pueden aplicar a un elemento dentro del código HTML por lo que se incumple un principio básico de cualquier lenguaje
      - →**REUTILIZACIÓN DE CÓDIGO**
    - Es la forma más rápida en Javascript para recorrer el DOM

```
#title{
    color: aqua;
}

#text{
    color: blueviolet
}
```

ESTILOS

- **4.- Sintaxis y tipo de selectores**
  - **Selector universal**
    - Recomendable únicamente para resetear los estilos del navegador
    - Con el * se hace referencia a todos los elementos HTML

```
/* Selector Universal */

*{
    color: red;
}
```

    - El resto de estilos tienen mayor peso o prioridad
    - No suele utilizarse ya que es complicado tener que aplicar el mismo estilo a todos los elementos de una página.

ESTILOS

- **4.- Sintaxis y tipo de selectores**
  - **Selectores Básicos / atributo**
    - Seleccionan los elementos que tienen ese atributo
    - El atributo se pone entre corchetes []
    - Se pueden seleccionar los que tienen un valor concreto en un atributo

```
[href]{
    color: green;
}

[href="#"]{
    color: red;
}
```

ESTILOS

- **4.- Sintaxis y tipo de selectores**
  - **Selectores Básicos / atributo**
    - Seleccionan incluso los que contienen al menos una palabra en el class con ~ o con concordancia exacta con |

```
/*Selecciona los que tienen contenido verde en el class*/
[class~="verde"]{
    color: green;
}

/*Selecciona los que tienen exactamente ese valor o que empiece
por el valor seguido de un guión*/
[class|="verde"]{
    color: blue;
}
```

ESTILOS

- **4.- Sintaxis y tipo de selectores**
  - **Selectores Básicos / atributo**
    - Con ^ seleccionamos los elementos que **empiecen** por el atributo que queramos
    - Con $ los que **terminen** por el atributo que queramos
    - Con * los que **contengan** el atributo que queramos

```
[class ^="rojo"]{
    color: purple;
}

[class $="rojo"]{
    color: steelblue;
}

[class *="rojo"]{
    color: brown
}
```

ESTILOS

- **4.- Sintaxis y tipo de selectores**
  - **Selectores Combinadores**
    - HERMANO ADYACENTE
      - Selecciona a un hermano que esté justo debajo y le aplica un estilo

```
h1 + h2{
    color: red;
}
```

ESTILOS

- **4.- Sintaxis y tipo de selectores**
  - **Selectores Combinadores**
    - HERMANO GENERAL
      - Busca hermanos que comparte un mismo padre, da igual el orden

```
/* Selector de hermano general*/
h1 ~ h3{
    color: green;
}
```

      - Aplica a lo que encuentre por debajo de él
      - Seleccionar los elementos que están por encima en CSS no se puede hacer (habría que recurrir a Javascript)

ESTILOS

- **4.- Sintaxis y tipo de selectores**
  - **Selectores Combinadores**
    - DESCENDENTES
    - Aplica estilo a los elementos que comparten un padre en común

```
/*Selector descendente*/

p span{
    color: red;
}
```

    - En el ejemplo se estaría dando estilo a los span que estén dentro de un p

ESTILOS

- **4.- Sintaxis y tipo de selectores**
  - **Selectores Combinadores**
    - HIJO DIRECTO

```
/*Selector de hijo directo*/

p > span{
    color: blue;
}
```

    - Aplica estilo únicamente al elemento que esté a distancia 1 de la jerarquía, es decir, si hubiera otro elemento entre medias no aplica

    - Si entre el p y el span hubiera algún elemento intermedio no aplicaría el estilo

ESTILOS

- **5.- PREPOS**
  - Los navegadores tienen códigos que se añaden a las propiedades experimentales (aún no aprobadas por W3C)

```
<!--[if IE]>
  El código aquí situado será leído únicamente por los navegadores Internet Explorer
<![endif]-->

<!--[if (IE 7) | (IE 8)]>
  Y el situado aquí,  sólo por Internet Explorer 7 o Internet Explorer 8
<![endif]-->
```

  - Esto es así hasta que se estandariza
  - Usaremos algún software que se encargue de generar los prefijos propietarios mientras nosotros creamos CSS plano

ESTILOS

- **5.- PREPOS**
  - Propiedades especiales (Chrome)
  - Cada navegador tiene su propio motor gráfico:
    - IE: Trident
    - Firefox: Gecko
    - Opera: Presto
    - Chrome y Safari: Webkit

Consultar buscadores especializados: Should I Prefix y Can I Use

ESTILOS

6.- HERENCIA Y CASCADA

- HERENCIA
  - La herencia se aplica con el valor **inherit**
  - Se obliga al elemento a heredar la propiedad de su elemento más cercano.
  - Cuidado porque algunas propiedades no se heredan
    - Los márgenes (es poco probable que un elemento hijo necesite los mismos que su padre)

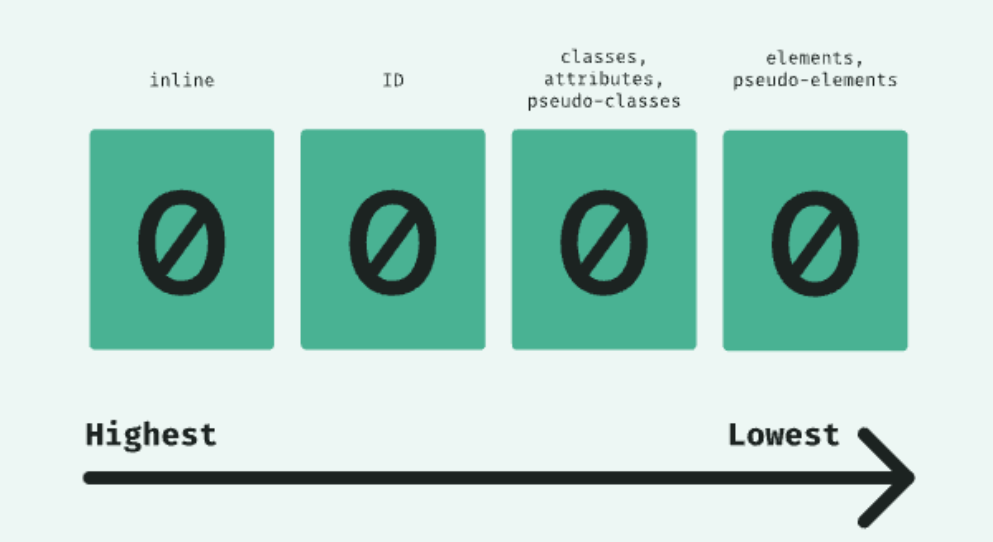
ESTILOS

- **6.- HERENCIA Y CASCADA**
  - CASCADA
    - Es el orden en el que se van aplicando estilos.
    - El lenguaje se lee de arriba abajo, por tanto, los estilos que vienen más abajo pueden **sobrescribir** los anteriores.
    - Hay conflicto de estilos muchas veces
      - Un mismo elemento tiene varios estilos para aplicarle
      - ¿Con cuál se queda?

```
h1 {
    color: red;
}
h1 {
    color: blue;
}
```

ESTILOS

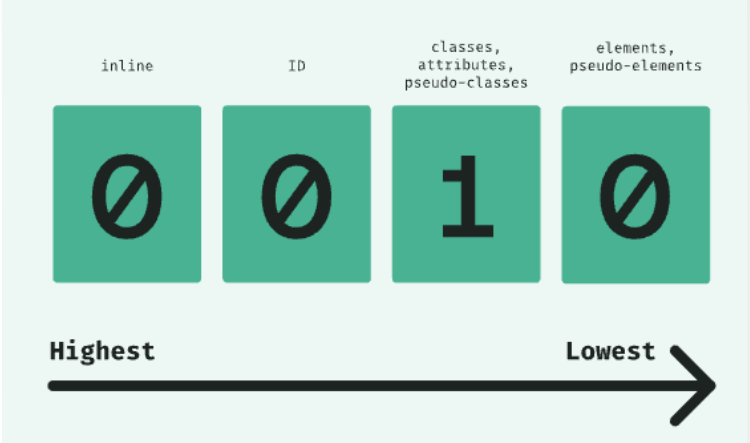
- **7.- ESPECIFICIDAD**
  - Cuando hay conflicto de estilo entra en juego la **ESPECIFICIDAD**
  - Determina que estilo tiene más prioridad
  - La W3C ha publicado un baremo con las prioridades a aplicar en caso de conflicto



ESTILOS

7.- ESPECIFICIDAD

```
.hello-header {
  color: red
}

div > div > h1 {
  color: blue
}
```



ESTILOS

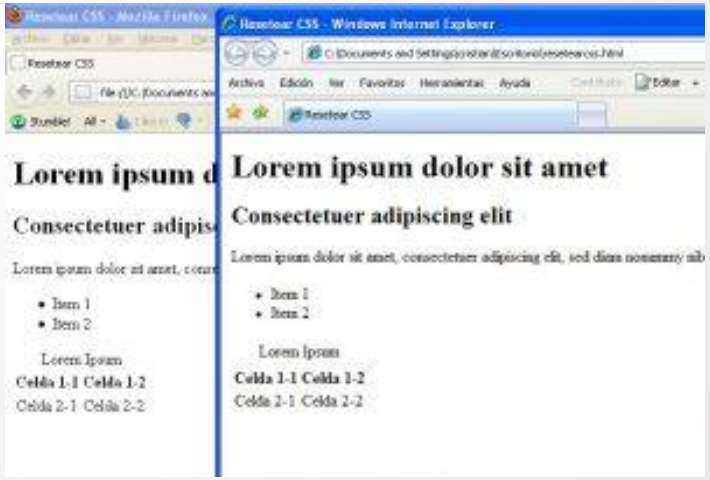
7.- ESPECIFICIDAD

- La regla **!important** gana las guerras de las especifidad en CSS
- Invalida el resto de reglas
- Está considerado como una mala práctica, cambia el modo en que suele funcionar la cascada → dificulta la depuración de problemas.

```
h1 {
  color: blue !important
}

h1 {
  color: yellowgreen !important
}
```

ESTILOS

- **8.- RESETEAR ESTILOS DE LOS BROWSER**
  - Los navegadores tienen sus propios estilos
  - Todos los navegadores tienen por defecto una hoja de estilos y en el caso de que se interprete html5 sin estilo aplican su estilo.
  - Se puede ver cuando pones un enlace → azul y subrayado
  - O cuando ponemos un párrafo → tiene márgenes que tu no has dado



ESTILOS

- **8.- RESETEAR ESTILOS DE LOS BROWERS**
  - Necesitamos sobrescribir esos estilos para que el código sea igual en todos los navegadores
    - → RESETEAR ESTILOS DE LOS NAVEGADORES
  - NORMALIZE.CSS
    - La más recomendada
    - Hay más, por ejemplo RESET.CSS → hándicap!
      - Pone a cero hasta los elementos de HTML y tendrías que volver a escribir el estilo de todo otra vez desde cero

ESTILOS

- **8.- RESETEAR ESTILOS DE LOS BROWERS**

```
<head>
    <title>RESETEAR ESTILOS DE LOS NAVEGADORES</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="normalize.css" rel="stylesheet">
    <link href="styles-dist.css" rel="stylesheet">
</head>
```

ESTILOS

- **9.- METODOLOGÍA DE ESCRITURA CSS - BEM**
  - Una metodología en CSS es una serie de consejos para estructurar nuestro código de forma sencilla, escalable y reutilizable.
  - Es decir, si metemos un elemento nuevo que no se desmaquete todo y nos obligue a reestructurar todo el css
  - Al usar una metodología simplemente añadimos los estilos del nuevo elemento y lo demás no se ve afectado

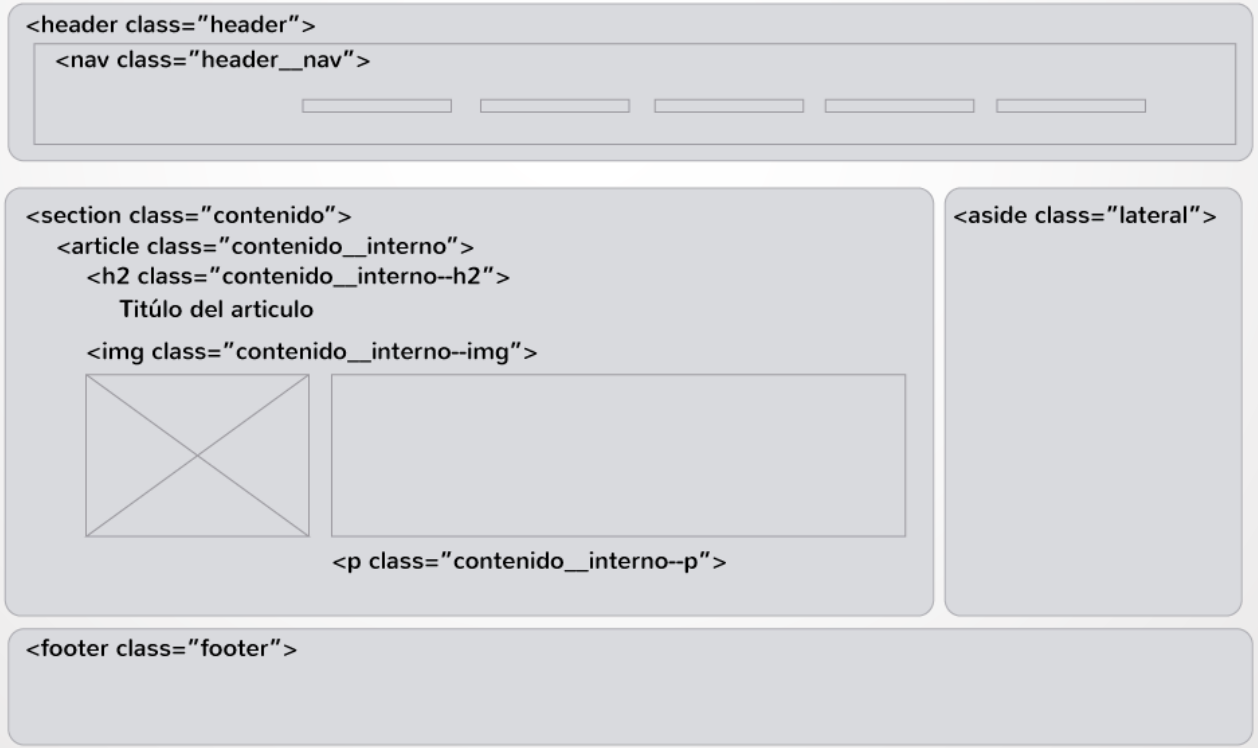
ESTILOS

- **9.- METODOLOGÍA DE ESCRITURA CSS - BEM**
  - Todo el estilo visual del sitio se maneja a través de clases (solamente usar identificadores con JavaScript)
  - Es una de las metodologías más usada en el mundo, son buenos consejos que ayudan a evitar conflicto de estilos.
  - Propone dividir la interfaz de usuario en bloques independientes para crear componentes.

ESTILOS

9.- METODOLOGÍA DE ESCRITURA CSS - BEM

- **BLOCK** → Cualquier elemento autónomo y aislado dentro de nuestro documento. Es donde se ubican los diversos elementos



ESTILOS

- **9.- METODOLOGÍA DE ESCRITURA CSS - BEM**
  - **ELEMENT** → Son las diversas partes que componen un bloque, y jamás existirían sin la definición previa de este.
    - Cada elemento nace a partir de la clase del bloque que conforma seguida de dos **guiones bajos __** y el nombre clave del elemento.

```
<header class="header">
    <!-- elemento 1 -->
    <img src="example.png" alt="logo" class="header__logo">
    <!-- elemento 2 -->
    <a href="example.html" class="header__link">example</a>
</header>
```

```
.header__logo {
    filter: brightness(0.01);
}
.header__link {
    color: #2e2e2e;
}
```

ESTILOS

9.- METODOLOGÍA DE ESCRITURA CSS - BEM

- **MODIFIER** → Es un bloque o elemento que se repite en otro lugar de nuestra web pero con alguna modificación.
  - Ejemplos: mismo formulario pero con distinto color de texto, tamaño de fuente, etc…
  - Un ejemplo típico es un menú de navegación cuando se pone en la parte inferior y se quiere modificar el color para adaptarlo a footer oscuros, es el mismo bloque repetido pero con alguna modificación.

ESTILOS

9.- METODOLOGÍA DE ESCRITURA CSS - BEM

- Para agregar una variación del mismo **bloque** o **elemento**, se utilizan dos **guiones medios --**

```
<header class="header">
   <!-- aquí van mis elementos -->
</header>
```

```
.header {
    background-color: #fff;
    color: #2e2e2e;
}
```

```
<header class="header header--dark">
   <!-- aquí van mis elementos -->
</header>
```

```
.header {
    background-color: #fff;
    color: #2e2e2e;
}
.header--dark {
  background-color: #2e2e2e;
  color: #fff;
}
```

ESTILOS

9.- METODOLOGÍA DE ESCRITURA CSS - BEM

```html
<header class="header header--dark">
    <!-- elemento 1 -->
    <img src="example.png" alt="logo" class="header__logo header__logo--light">
    <!-- elemento 2 -->
    <a href="example.html" class="header__link header__link--light">example</a>
</header>
```

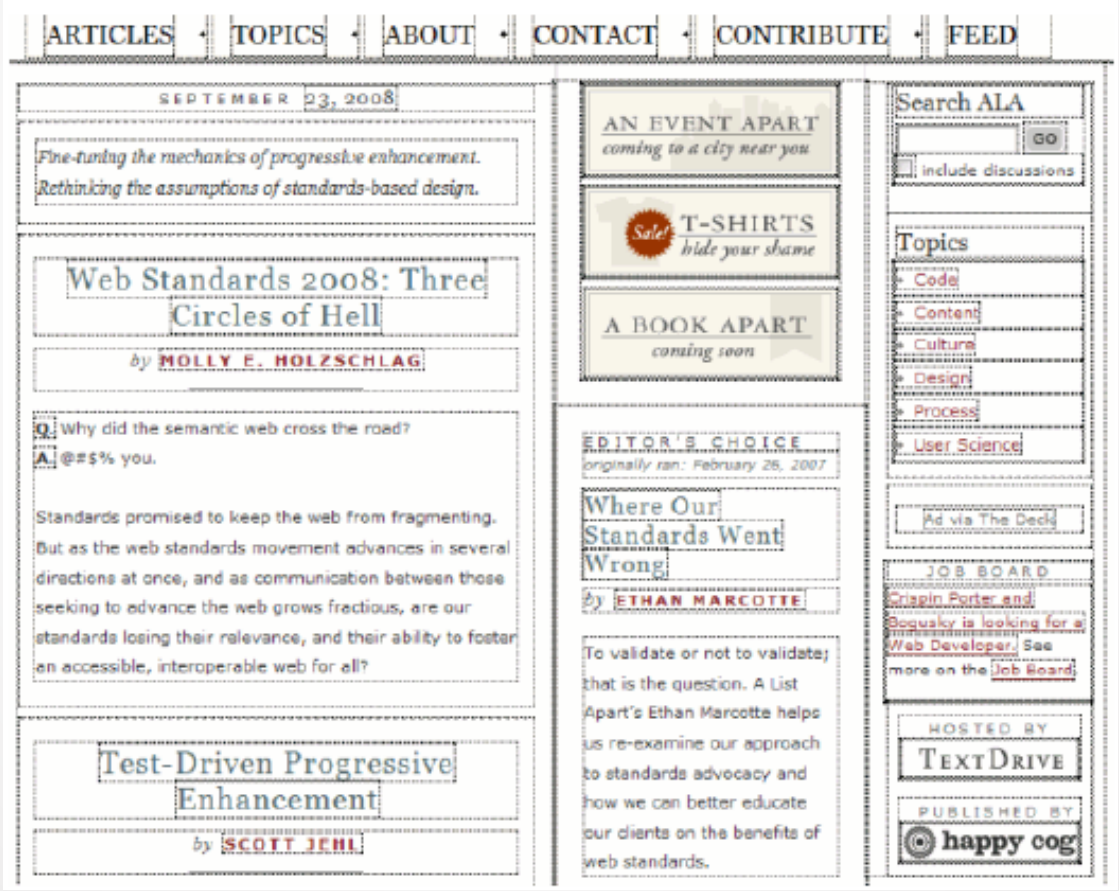
```css
.header__logo {
    filter: brightness(0.01);
}

.header__logo--light {
    filter: brightness(100);
}

.header__link {
    color: #2e2e2e;
}

.header__link--light {
    color: #fff;
}
```

ESTILOS

- **10.- BOX-MODEL (Caja Contenedor)**
  - El modelo de cajas es la característica más importante de CSS, condiciona el diseño de todas las páginas web.
  - Todos los elementos incluidos en una página HTML se representan mediante cajas rectangulares.
  - La forma en el que el navegador dibuja esas cajas en pantalla → LAYOUT
    - Si un elemento está al lado de otro, si se pone debajo, si ocupa la mitad de la pantalla…

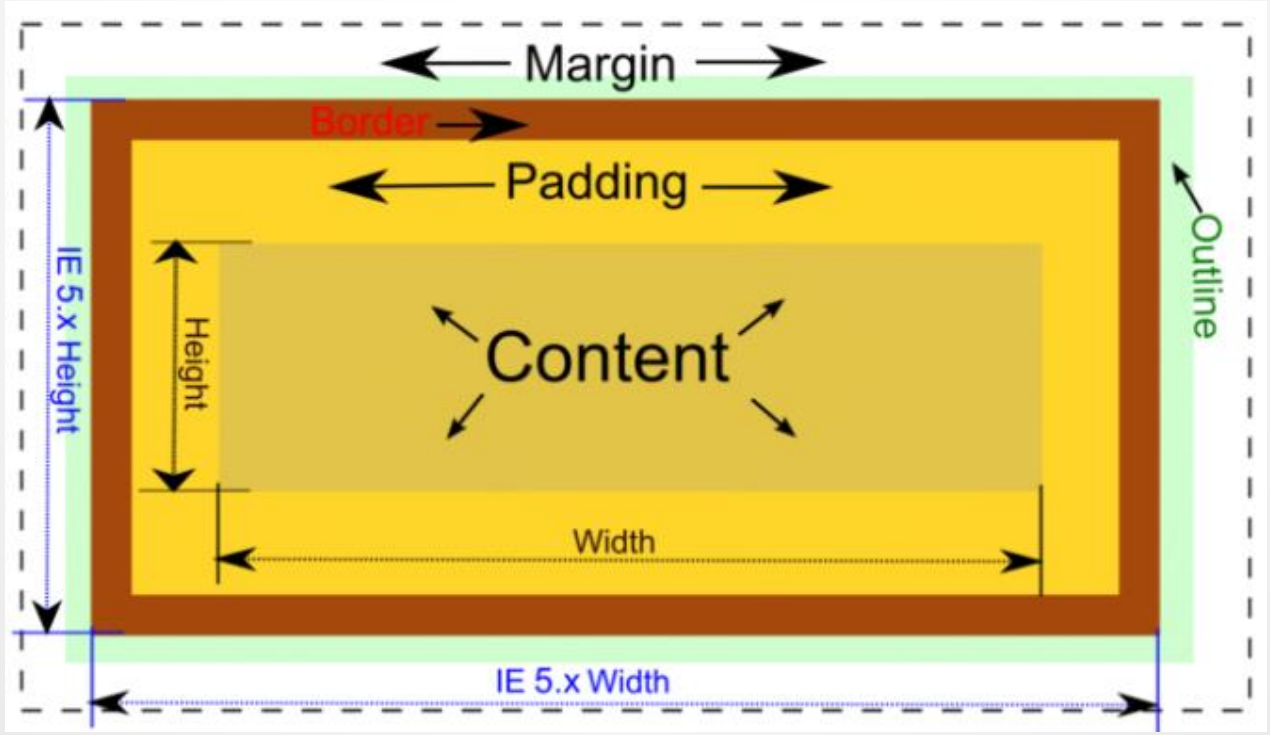
ESTILOS

- **10.- BOX-MODEL (Caja Contenedor)**



ESTILOS

- **10.- BOX-MODEL (Caja Contenedor)**
  - Las propiedades principales de cada una de estas cajas son **width** (ancho), **height** (alto), **padding** (relleno), **border , margin** y **content.**

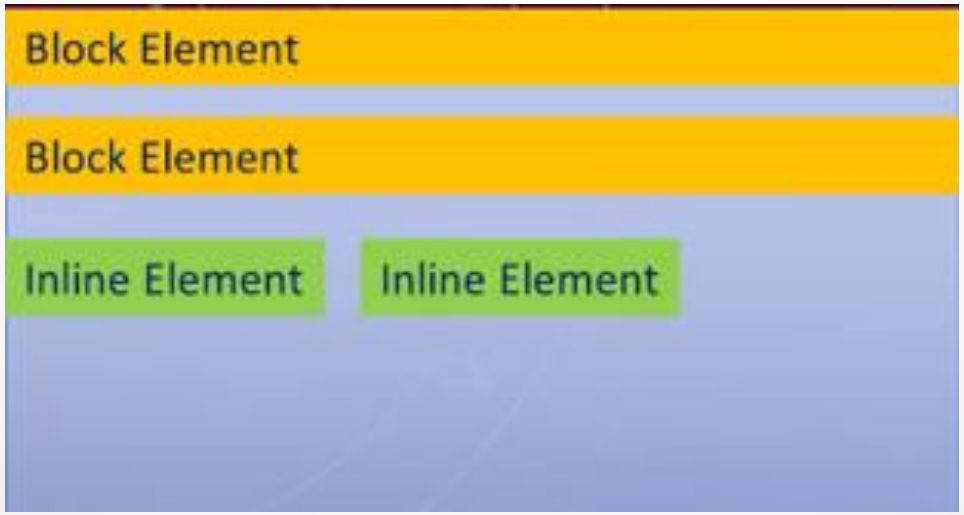


ESTILOS

- **10.- BOX-MODEL (Caja Contenedor)**
  - Hay 2 tipos de elementos HTML:
    - **Elementos en línea (inline)**
      - Son elementos que solo ocupan su contenido
      - No se puede modificar ni su ancho ni su alto
      - Ejemplo: <a>
    - **Elementos de bloque (block)**
      - Ocupan todo el ancho disponible
      - Se les puede asignar ancho y alto
        - Por defecto ancho es todo el espacio disponible y alto lo que ocupe su contenido

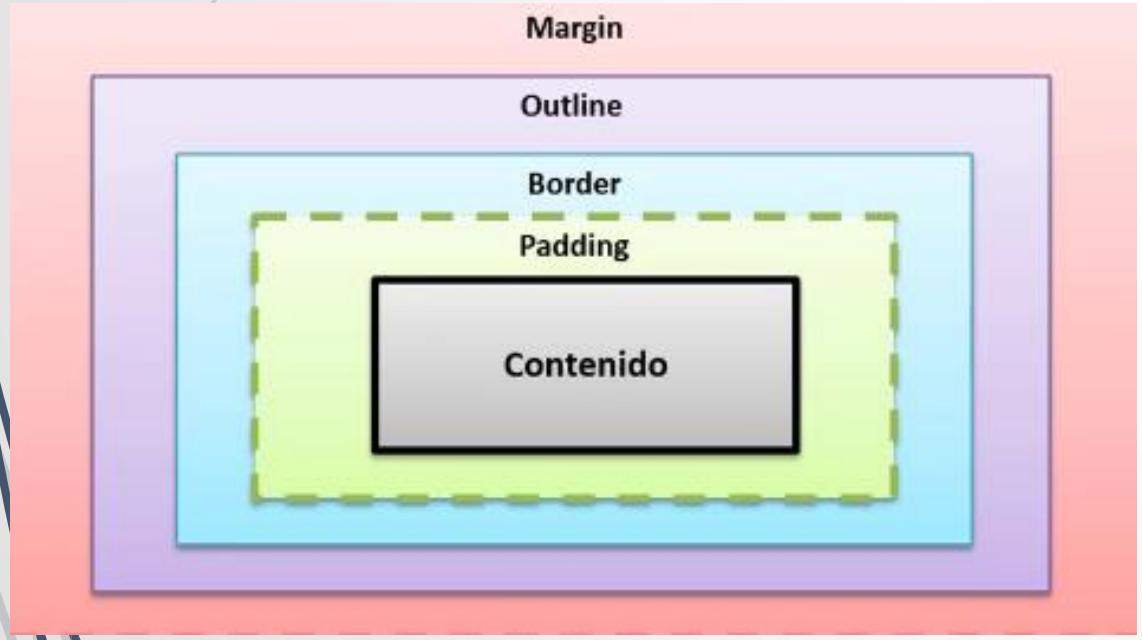
ESTILOS

- **10.- BOX-MODEL (Caja Contenedor)**
  - Estas propiedades se pueden modificar con el atributo **display** (lo vemos más adelante)



ESTILOS

- **10.- BOX-MODEL (Caja Contenedor)**
  - En el inspector del navegador veremos este modelo:



padding → Distancia entre contenido y borde

outline → Se dibuja por fuera del borde

margin → Distancia entre un elemento y otro