Apuntes: Hojas de Estilo (I)

DAW- IES Doñana - 2º DAW 21/22

Eduardo Martínez Romero

1. Fundamentos de CSS

CSS (*Cascade Style Sheets*) es un lenguaje de hojas de estilo, el cual:

- Aplica estilos de manera selectiva a ciertos elementos de nuestro HTML. Tambie hay atribytos a los que se les puede aplicar estilo
- Permite adaptar un sitio web para diferentes dispos
- Principal ventaja, permite el mantenimiento y la reusabilidad.

1.2 Estilo

Conjuto de reglas que debe aplicar el navegador para mostrar la información con cierta apariencia.

1.3 Estructura

Su estructura se basa en selectores que, entre llaves, contienen propiedades a las cuales asignamos un valor:

```
selector {
    propiedad: valor;
}
```

2. Vincular hoja de estilo

2.1. Fichero independiente

Por medio de la etiqueta `link`, el atributo `rel` para indicar el tipo documento que vamos a usar (`css`) y tambiñen el atributo `href` para indicar la ubicación de nuestro fichero css.

```
<link rel="stylesheet" href="styles.css">
```

2.2. La etiqueta `style`

Esta etiqueta sólo puede ir definida en el `head`, y aplicará el código css que vaya en su interior:

```
<head>
    <style>
        body {
            background: black;
        }
    </style>
</head>
```

3. Selectores

Los selectores son los elementos que definen el estilo de las distintas propiedades que definamos, asignandoles un valor

```
/* Estos son selectores de elementos básicos.*/
selector {
    propiedad1: valor;
    propiedad2: valor;
    propiedad3: valor;
}
```

3.1 Aplicación de estilos en cascada

Un mismo elemento puede estar sujeto a varias hojas de estilo CSS. Existen normas que indican de forma inequivoca cuál de todas las instrucciones es la que debe afectar a un determinado elemento HTML.

3.1.1 Reglas de prioridad

Tenemos 3 reglas que determinan la prioridad:

1. Importancia (navegador < autor < usuario).
2. Especificidad (style inline > #id > .clase > elemento)
3. Orden de aparición (El último en aparecer sobrescribe a los anteriores):
   a). Estilo definido en páginas externas.
   b). Estilo definido en la cabecera.
   c). Estilo definido en línea (Usando el atributo `style` dentro de la etiqueta).

> Podemos hacer saltar esta cadena de prioridades usando el modificador `!importat` tras el valor de la propiedad.
> El uso de `!important` esta completamente desaconsejado.

3.2. Selectores de etiqueta

Especifica la etiqueta HTML a la que vamos a aplicar los estilos.

```
p {
    color: #f00;
    font-style: italic;
    font-weight: bold;
}
h2 {
    color: #00f;
    font-size: .2em; /* i.e. 0.2em */
}
```

3.3 Selectores de clase

Para dar estilo a varios elementos que tengan la misma clase definida dentro del atributo HTML `class`.

```
.miClase {
    color: #999;
    border: 1px solid black;
}
```

Delante del `.` punto podemos limitar el alcance de la clase:

- `.miestilo` no es lo mismo que `*.miestilo`.
- `p.miestilo` no es lo mismo que `p, .miestilo` (El segundo es un selector de grupo, compuesto por el selector de etiqueta `p` y el selector de clase `miestilo`).

3.4. Selectores de ID (#)

Aplica el estilo por al elemento definido dentro del atributo `id` de HTML.
A diferencia de las clases, los id's son más exclusivos, ya que sólo se pueden aplicar a un elemento de todo el código.

```
#tabla {
    background-color: #0f0;
    border-color: #f00;
}
```

3.5. Selectores de grupo

Aplica estilo a un conjunto de etiquetas y clases diferentes, separadas por coma.

> Tratar de usar de forma coherente

```
h1,
h2,
h3 {
    color: purple;
}
```

3.6 Selector universal

Se aplicará a todos los elementos de la página, usando `*`asterisco.

```
* {
    font-family: "Times New Roman", Times, serif;
}
```

3.7. Selectores descendientes

Por definición del efecto cascada: Todos los selectores anidados heredan los valores de las propiedades asignadas al selector exterior.

```
<html>
<head>
    <style>
        #estilo1 {
            color: f00;
            text-align: center;
        }
    </style>
</head>
<body>
    <div id="estilo1">
        <!--p y span heredan de su div padre -->
        <p>Hola <span>Mundo!</span>.</p>
    </div>
</body>
</html>
```

- En función del contexto podemos referirnos a elementos que se encuentren dentro de otros.
- El estilo no afectará a las etiquetas que no se encuentren dentro del elemento padre.
- Se pueden combinar diferentes tipos de selectores.

```
/* Todo elemento 'a' hijo de un elemento 'li' */
li a {
```

```
    color: f00;
}
```

3.8 Selectores hijos (`>`)

Selecciona sólo al hijo directo del elemento.

```
/* Todo elemento 'a', hijo directo de cuerpo */
body > p {
    color: f00;
}
```

3.9 Selectores de hermanos (`+`)

- Aplica el estilo a un elemento que esté justo después.
- Ambos tienen el mismo padre.
- No afecta a los siguientes hermanos

```
/* Todo elemento 'h2', y hermano 'p' */
h2 + p {
    color: f00;
}
```

3.10 Selectores de atributo

Selecciona elementos HTML en funcion de sus atributos y/o los valores de sus atributos. Para mejorar la precisión de nuestras selecciones hacemos uso de **expresiones regulares**.

3.10.1 Expresiones regulares

Símbolo	Definición
=	Valor excato
~=	Contiene la palabra.
|=	contiene "valor" o "valor-(...)" -seguido de un guión-
^=	comienza por
$=	termina por
*=	contiene

```
/*
    Todo elemento 'img' cuyo valor del atributo 'src'
    termine en '.png'.
*/
img[src$='.png'] {
    border: 1px solid red;
}
```

21/10/2021 15:25

Pseudo-clases

Estados de elementos (`:`)

- Define estilos sobre estados de elementos.

Estados para el elemento `a`:

- `:link`
- `:visited`
- `:hover`
- `:active`

> Existen multitud de pseudoclases, que afectan a otros elementos HTML.

Estados de elementos de formularios:

- `:in-range`
- `:outof-range`
- `:default`
- `:invalid`
- `input[type="radio"]:default`

Ejemplos de pseudoclases:

```
a:link {
    color: indigo;
}

a:visited {
    color: lightblue;
}

a:hover {
```

```
    color: orange;
}
```

Exclusiones (`:not()`)

- Aplica a todos los elementos que no cumplan la condición.
- Se puede usar con elementos `id` o `class`.

```
/* Todo "p" que no sea de clase ".clase"*/
p:not(.clase) {
    color: magenta;
}
```

Pseudo-elementos (`::`)

- Complementa a los selectores y pseudoclases para aplicar estilos a elementos especiales:
  - `::fist-line` (Para `p`, `h1`, `td`...).
  - `::first-letter` (Para `p`, `h1`, `td`...).
  - :`:before` & :`:after` (con la propiedad `content`).
  - `::selection` (Apariencia cuando un usuario selecciona texto).
  - `::placeholder` (Experimental. Modificar texto de ejemplo incluido en elementos como `inputs` o `textareas`).
  - `::marker` (Estilo de los elementos de listas).

```
h2:before {
    content: url(icono.png);
}
```

Propiedades abreviadas

Se pueden declararar simultáneamente varios valores a una propiedad.

Por ejemplo la propiedad `padding` está compuesto a su vez de las propiedades:

- `margin-top`
- `margin-bottom`
- `margin-left`
- `margin-right`

Podemos definir todas esas propiedades en una sola linea, usando la propiedad abrevoada`margin` y pasando cada valor, serparado por comas.

> En ocasiones influye el orden de los valores que asignamos. Algunas etiquetas no funcionarán como esperamos si no respetamos dicho orden.

Para el caso de `margin`, dIsponemos de cuatro métodos para aplicar, y su aplicación se determinará por el número de valores que se definan, como se puede ver a continuación:

Nº valores	Resultado
1	Mismo valor a todas la propiedades.
2	1. top & bottom 2. left & right
3	1. top 2. left & right 3. bottom
4	En sentido horario (top,right,bottom,left)

27/10/2021 15:33

---

4. Herencia

En concepto general, consiste en que los elementos ascendentes permiten que los descendientes puedan adquirir las propiedades de su padre. Pero esta herencia puede ser modificada según nuestro propótiso.

¿Todas las propiedades se heredan?

No todas las propiedades en css se heredan, por ejemplo:

```
<style>
    p {
        border: 1px solid black;
        color: red;
    }
</style>
<body>
    /* 'span' hereda el color de 'p', pero no su borde*/
    <p>Parrafo con <span>span</span> y <a>enlace</a>.</p>
</body>
```

En el siguiente enlace se pueden consultar todas las propiedades que disponemos en CSS. También podremos ver cuáles se heredan por defecto, y cuáles no.

CSS Property Index [w3.org]

Recopilación de propiedades CSSque no se heredan

- `border-collapse`
- `border-spacing`
- `caption-side`
- `color`
- `cursor`
- `direction`
- `empty-cells`
- `font-family`
- `font-size`
- `font-style`
- `font-variant`
- `font-weight`
- `font-size-adjust`
- `font-stretch`
- `font`
- `letter-spacing`
- `line-height`
- `list-style-image`
- `list-style-position`
- `list-style-type`
- `list-style`
- `orphans`
- `quotes`
- `tab-size`
- `text-align`
- `text-align-last`
- `text-decoration-color`
- `text-indent`
- `text-justify`
- `text-shadow`
- `text-transform`
- `visibility`
- `white-space`
- `widows`
- `word-break`
- `word-spacing`
- `word-wrap`

¿Todas las etiquetas se heredan?

No todas las etiquetas en css heredan.

> TODO: Agregar ejemplo de etiqueta que no se hereda

inherit /initial

Estos valores especiales nos permiten declarar de forma explícita que queremos que se herede el valor de la propiedad padre, o en su defecto se aplique el estilo por defecto de esa etiqueta.

> Efectivamente, existe una "plantilla inicial" de CSS que el navegador aplica por defecto unos determinados estilos a los distintos elementos del documento. Por ejemplo `h1` ó `codeblock`.

Inherit

Forzamos a un elemento a heredar las propiedades de su padre.

```
/* Con esto forzamos que el enlace cambie de color */
p {
    color: red;
    border: 1px solid black;
    margin: 10px;
    padding: 50px;
}

/* Tambien haremos que tenga borde */
a {
    color: inherit;
    border: inherit;
}
```

Initial

Forzamos a un elemento a cargar sus propiedades iniciales.

```
/* Con esto forzamos que el enlace cambie de color*/
p {
    color: red;
    border: 1px solid black;
    margin: 10px;
    padding: 50px;
}

/* Tambien haremos que tenga borde*/
a {
    color: inherit;
    border: inherit;
}
```

5. Reset

La finalidad es que el punto de partida para todos los navegadores [... ni idea pare XD]

Modelo de contenedor

- Todas las etiquetas son consideradas como cajas o contenedores.
- Dentro de cada caja se muestra el contenido.
- Bordes y relleno son transparentes.

Elementos en bloque

- Ocupa todo el espacio de su contenedor.
- Pueden contener elementos en bloque y de linea.
- Ejemplos: Tablas, listas, elementos `div`, cabeceras, párrafos...

Elementos en línea

- Ocupa todo el espacio de su contenido.
- Puede contener otros elementos en línea.
- Ejemplos: `strong`, `em`, `img`, `a`...
- Tienen `margin` y `padding`, pero sólo a su derecha e izquierda.
- Ignoran las propiedades `height` y `width`.

```
a {
    margin: 20px 40px 30px 10px; /* i.e: 0 40px 0 10px*/
    padding: 20px 40px 30px 10px; /* i.e: 0 40px 0 10px*/
}
```

Estructuras de página

- `float`: Flotar la caja de un elemento de izquierda a derecha (`right`, `left`, `none`).
- `display`: Nos marca y fuerza cómo vamos a mostrar un elemento (`block`, `inline`, `none`, ...

> La propiedad `display: none` hará que dicho elemento no se visualice.

> Práctica: Hacer que no se muestre un elemento, omitiendo el espacio que ocupa, y otro que tampoco se muestre pero si conserve su espacio.

- `clear`: Elementos justamente detrás de un elemento flotante. (Puede estar o no junto a elementos flotantes).
- `position`: Lo veremos la prox. semana (`top`, `right`, `bottom`, `left`, `relative`, `absolute`, `fixed`, `static`)