CSS

CONTENIDO

INTRODUCCIÓN

CSS es un lenguaje de hojas de estilo utilizado para controlar el aspecto de los documentos HTML.

Separar la definición de los contenidos y la definición de aspecto obliga a crear documentos bien definidos (semánticos), mejora la accesibilidad, facilita el mantenimiento y permite visualizar el documento en diferentes dispositivos.

Estándar CSS

Actualmente se está trabajando para alcanzar una versión CSS 3 que aúne el actual CSS 2.1 y los diferentes módulos que han ido surgiendo y siendo aceptados por la comunidad entre estas dos versiones.

Hasta que se consiga esto, el panorama actual es complicado, pues existen diferentes documentos (CSS Colores…) que llevan distinto ritmo (uno puede ir por la versión definitiva y otro por el borrador) y los diferentes navegadores van incorporando propiedades nuevas sin consenso previo (así, Google puede decidir darle soporte a una propiedad nueva y Mozilla a otra). Esto nos complica la vida como desarrolladores, pues debemos conseguir que nuestra web se vea en todos los navegadores -tanto en las versiones más recientes como en las de los últimos años (debe ser compatible con versiones antiguas y debe tener en cuenta que los navegadores se actualizan mucho hoy en día).

Por esto, es imprescindible conocer el soporte de CSS que tienen los navegadores más populares. La parte del navegador que interpreta HTML y CSS para mostrar las páginas se denomina motor (*layout engine*). La versión del motor es más importante que la del navegador.

Los estados por los que pasa una nueva norma CSS antes de ser oficial son: borrador, recomendación, testing, estándar.

FORMAS DE APLICAR ESTILOS A UN DOCUMENTO HTML

1) Mediante un enlace a un documento CSS externo en la cabecera (vía link)

Es la forma más correcta de hacerlo. Se crean uno o varios archivos CSS y se enlazan con el documento HTML mediante sentencias que aparecerán en este, dentro de la cabecera (*head*) del documento:

```
<link rel="stylesheet" href="/css/estilos.css" media="screen">
```

Este elemento puede contener tan solo el atributo *rel* (tipo de relación entre el archivo css y la página html, será siempre *stylesheet*) y la ruta al documento enlazado (*href*) o también el medio para el que

se aplicará este documento (*all*/*screen*/*print*/*handheld*) y/o el tipo de recurso enlazado (que será *type="text/css"* , aunque este ya no es necesario en HTLM5 -se recomienda no usarlo).

Lo que se hace muchas veces es tener dos ficheros css. Esto es interesante para un sitio web con diferentes departamentos: para la web en general tendré un estilo global, pero luego puedo tener algún departamento que quiera personalizar.

2) Mediante un enlace a un documento CSS externo en la cabecera (vía style)

Dentro de la cabecera creo un elemento *style* donde escribo reglas CSS: esto sirve para hacerle un parche al css enlazado pero no es recomendable, lo ideal es hacerlo todo en el documento CSS.

```
<style type="text/css" media="screen">
@import '/css/estilos.css';
</style>
```

O también (otra forma cada vez menos usada):

```
<style>
@import url("css/estilos.css");
</style>
```

3) Mediante estilos en la cabecera (style)

```
<style type="text/css">
h1 {color: red; font-family: Arial; font-size: large; }
</style>
```

4) Mediante reglas aplicadas directamente al elemento

```
<p style="color: black; Font-family: Verdana;">texto del párrafo</p>
```

5) Mediante reglas aplicadas directamente al texto

```
<h1><font color="red" face="Arial" size="5">Titular</font></h1>
```

PARTES DE UNA REGLA CSS

Una regla es cada estilo que se aplica a un elemento (el color de fuente, el tipo, el tamaño, etc.).

El selector es lo que está a la izquierda de la llave de apertura y me indica a qué elementos le voy a aplicar los estilos.

La declaración es el conjunto de estilos que se aplican a un tipo de elemento concreto (que estará indicado mediante el selector). Tiene el siguiente formato: formato "propiedad: valor; valor; valor;"

La propiedad es la característica que se modifica en el elemento, y el valor el nuevo aspecto que va a presentar esta propiedad.

La forma más correcta de escribir las reglas CSS es cada una en una línea y con sangría. Hay herramientas para reducir los espacios extra cuando sea necesario (por ligereza), pero en principio lo ideal es escribir un código legible y claro. Ejemplo:

```
h2{
  border-bottom: 2px green dotted;
  color: blue;
}
```

Se pueden incluir varias reglas dentro de un selector y también se pueden poner varios selectores para una regla (utilizando comas).

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

COLISIONES

En ocasiones el navegador se encontrará con contradicciones al interpretar el código de una página web. Hay una serie de normas que le permiten saber a cuál debe darle prioridad:

1. Según la importancia de la regla: si lleva la palabra clave **!important** supera a las demás.

2. Según su origen:

    2.1. Usuario (las reglas que este establece directamente sobre la página al visualizarla): hay navegadores que permiten que el usuario le aplique estilo a las páginas que va a ver (esto suele estar enfocado a accesibilidad: tamaño de letra, colores de fondo desactivados, etc.)

    2.2. Diseñador (las reglas indicadas por el desarrollador web)

        2.2.1. Atributo de estilo (“style”) que forma parte del elemento en cuestión

        2.2.2. Elemento “style” en la cabecera (head)

        2.2.3. Hoja externa CSS (enlazada en el HTML)

    2.3. Navegador (hoja de estilos por defecto)

3. Según la especificidad del selector: el más concreto será el que prevalece.

Un método para conocer la especificidad de un selector con respecto a otro es el método ABC, donde el número mayor tiene prioridad; el número correspondiente a cada selector se calcula así:

A = número de selectores de id (que contiene nuestro selector)

B = número de selectores de clase, selectores de atributo y pseudo-clases

C = número de selectores de tipo y pseudo-elementos

Se ignora el selector universal.

Una vez hallados estos tres números, simplemente hay que poner uno detrás de otro hasta obtener un número de 3 cifras y comparar los números obtenidos (el mayor tendrá prioridad).

Ejemplos:

"section article p" es más específica que "p"

Aunque lo pueda parecer, "margin-right" NO es más específica que "margin", al igual que "border-style" NO es más específico que "border".

4. Según el lugar en que aparece la regla: dentro de la hoja de estilos o del apartado del HTML donde se añaden los estilos o se enlazan las hojas de estilo, el elemento que aparezca más abajo será el que tiene prioridad sobre todos los anteriores.

```
Para hacer una excepción en un borde:
border: 2px dotted red;
border-bottom-color: green;
```

(Si hay una contradicción entre el doc css enlazado y el elemento style de la cabecera, se interpreta lo último que lee el html: si link aparece antes de style, se aplica lo que ponga style por encima de link; si link aparece después de style, se aplica lo que ponga el css.)

```
<head>
        <meta charset="UTF-8">
        <title>Introducción a CSS</title>
        <link rel="stylesheet" href="css/introCSS.css">
        <style>
                h2{border-bottom: 1px green dotted;}
                h1{color:yellow;}
        </style>
</head>
```

MEDIDAS

Se indica el valor numérico seguido de la unidad (sin espacio intermedio). Si el valor es 0, no se pone unidad; si el valor es distinto a 0 y no se pone unidad, la medida se ignora; si el valor está entre 0 y 1, se puede omitir el 0 a la izquierda de la coma.

Hay medidas absolutas (siempre miden lo mismo) y relativas (cuyo valor se halla en función de otra medida, como la resolución o el monitor). El inconveniente de las unidades absolutas es que son poco flexibles (no se adaptan a diferentes medios). Se recomienda el uso de medidas relativas, pues mejoran la accesibilidad y permiten que los documentos se adapten a cualquier medio; normalmente se utilizan píxeles y porcentajes para el *layout* y em y porcentajes para el tamaño de letra.

Los navegadores, por defecto, muestran el texto con un tamaño de 16px (si queremos cambiar esto, debemos aplicar el tamaño de fuente al "body").

Unidades absolutas

· in (1 pulgada = 2,54cm)

· cm

· mm

· pt (1 punto = 1/72 pulgada = 0,35 mm), se suele usar para el medio "print"

· pc (1 pica = 12 ptos = 4,23 mm)

Unidades relativas

· em (1em viene a ser la anchura de la letra "M" de la fuente en cuestión), es relativa respecto al tamaño de letra del elemento, se utiliza para diseños que mantengan las proporciones (.1em= 0.1em)

· ex (1ex viene a ser 1/2 em o la altura de la "x" de la fuente del elemento), es relativa respecto al tamaño de letra del elemento

· px (píxel), es relativa respecto de la resolución de pantalla, se utiliza para una precisión total

p{width: 600px;} hace que los párrafos lleguen hasta los 600 px (tengan espacio blanco a la derecha de esa sangría -si están justificados a la izquierda)

· % (porcentaje), es relativa respecto del elemento padre, se utiliza para diseños líquidos o fluidos (si hago más pequeña o más grande la ventana del navegador el tamaño del elemento cambiará para mantener el porcentaje indicado) y sobre todo para definir la anchura.

p{width: 30%;} hace que mis párrafos tengan un 30% de anchura con respecto a la anchura del elemento contenedor

Ejemplo de herencia de una unidad relativa de medida (hay que tener en cuenta que se heredan los valores ya calculados):

```
body  {
        font-size: 12px;
        text-indent: 3em;
}
h1 {
        font-size: 15px;
/* El      text-indent      será (3em * 12px), lo mismo que el text-indent del body. NO 3x15. */
}
```

SELECTORES

SELECTOR UNIVERSAL (*)

Selecciona todos los elementos de la página: *

SELECTOR DE TIPO O ETIQUETA

Selecciona los elementos cuyo tipo se indica. Se pueden agrupar para seleccionar varios tipos de elementos diferentes en una regla (separados por coma y espacio).

h1, p, a {…}

SELECTOR DESCENDENTE ( )

Selecciona los elementos que se encuentran dentro de un cierto elemento: las diferentes partes de este selector deben estar separadas por un espacio, el último selector indica el elemento sobre el que se aplican los estilos y los anteriores forman la "ruta" donde se debe encontrar este elemento.

p * a {…} indica los enlaces que estén dentro de otro elemento que esté a su vez dentro de un párrafo

section p {…} indica los párrafos que estén dentro de una sección

section article p { … } se aplicará solo a los párrafos de artículos dentro de secciones.

SELECTOR DE CLASE (.)

Selecciona los elementos que pertenecen a una clase concreta (elemento HTML "class").

.destacado {…} selecciona los elementos que tienen el atributo class="destacado"

No hay que abusar de las clases: usarlas solo cuando no se puede usar otro selector.

SELECTOR DE ID (#)

Permite seleccionar un único elemento de la página (ya que solo un elemento de cada página puede tener un id concreto) a través de su valor del atributo id. Selectores comunes: contenido, noticias, anuncios.

#contenido {…} selecciona el elemento que tenga el id="contenido"

SELECTOR DE HIJOS (>)

Selecciona un elemento que es hijo directo (es decir, no es un nieto -hijo de un hijo) de otro elemento. Se utiliza, por ejemplo, si le quiero aplicar una característica a los párrafos que están dentro de la sección, pero no dentro de la sección y del artículo que está dentro de la sección.

section **>** p {…} significa "aplícale esto a los párrafos (no a los h1 u otra cosa) que son hijos directos de la sección, no los que están dentro de un artículo dentro de una sección.

section > * {} se aplica a todo lo que está dentro de la sección

SELECTOR DE HERMANO MENOR (~)

elemento1 **~** elemento2 {…} selecciona el elemento2 cuando está precedido por el elemento1 (ambos tienen el mismo padre, pero no tienen que estar justo a continuación uno del otro)

SELECTOR ADYACENTE (+)

elemento1 **+** elemento2 {…} selecciona el elemento2 cuando va justo a continuación del elemento1 (son hermanos y no puede haber nada entre el final del primero y el principio del segundo)

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

SELECTOR DE ATRIBUTOS

Selecciona elementos por sus atributos o valores de atributo.

Siempre que sea un selector de atributos lo que nos encontramos (lo primero) son corchetes. El punto es un selector de clase, la almohadilla un selector de id y el corchete un selector de atributo.

[x] selecciona los elementos que contenga el atributo “x”. [href] { border: 2px solid red; } (esto le pone un borde rojo a cualquier elemento que sea un enlace (dibuja un recuadro rojo alrededor de las palabras)

[x=valor] selecciona los elementos cuyo atributo “x” tenga el valor indicado. img[width=“16px”]

[x~=valor] selecciona los elementos cuyo atributo “x” tenga al menos el valor indicado. [title~="pendiente"] (alguna de las palabras del título es “pendiente” -con espacios alrededor)

[x|=valor] selecciona los elementos con este atributo y cuyo valor sea una serie de palabras que comienzan por el valor indicado (es útil sobre todo para el lenguaje de la página –“lang”) p[lang|="es"] {…} (todos los párrafos con atributo “lang” que tenga uno de los valores siguientes: es, es-ES, es-AR, etc.) (al ponerle la barra delante del igual coge cualquier valor de "lang" que empiece por "es")

[x^=valor] selecciona los elementos cuyo atributo “x” comienza por el valor indicado. a[href^="ftp://"] (antes del igual se pone un circunflejo)

[x$=valor] selecciona los elementos cuyo atributo “x” acaba por el valor indicado (esto es útil para destacar enlaces a determinado tipo de archivo, por ejemplo los .pdf) a[href$=".pdf"]

[x*=valor] selecciona los elementos cuyo atributo “x” contiene el valor indicado. img[alt*=logo]

COMBINACIÓN DE SELECTORES

Hay que tener mucho cuidado al combinar selectores, ya que un espacio de más o de menos cambia completamente el significado del selector.

h1 + h2 + h3 {...}

header h1+p {…} indica los párrafos que estén detrás de un h1 dentro de la cabecera.

header h1+p.novedad {…} indica los párrafos de la clase novedad que estén después de un h1 que esté dentro de un header

p.destacado {…} (combinación de selector de tipo y de clase: indica los párrafos de la clase “destacado”)

p .novedad{ … } se aplica a los elementos html de la clase novedad que estén dentro de un párrafo.

.novedad.importante {…} combinación de selectores de clase: indica los elementos que son de la clase novedad y de la clase importante

section#noticias {…} combinación de selector de tipo y de id: indica las secciones que tengan el id="noticias" (pensando en un sitio web con varias páginas)

section article#noticias {…} para que se aplique a los artículos de id noticias que estén dentro de una sección

section article #noticias {…} para un elemento que tenga el id noticias que esté dentro de un artículo y ese artículo dentro de una sección

p #aviso {…} indica los elementos con atributo id=”aviso” dentro de un elemento “p”

p, #aviso {…} indica los elementos “p” y los elementos con id=”aviso”

.aviso .especial {…} indica los elementos de clase especial dentro de un elemento de clase aviso

div.aviso span.especial {…} indica los elementos “span” de la clase “especial” que estén dentro de un elemento “div” de clase “aviso”

ul#menuprincipal li.destacado a#inicio {…} indica los enlaces que tengan el id=“inicio” y que estén dentro de un li de class=“destacado” que esté a su vez dentro de una lista ul con id=“menuprincipal”

[title*=“pendiente”] {…} indica los elementos cuyo atributo “title” contenga la palabra “pendiente”

[href^=”https://”][href$=”.pdf”] {…} indica los elementos con atributo “href” que empiece por “https://” y acabe por “.pdf”

a:link:hover {…} indica un enlace no visitado sobre el que paso el puntero

PSEUDO-SELECTORES

Son unos selectores especiales que permiten aplicar estilos a ciertas partes de un texto.

::after / **::before** permite añadir texto detrás o delante de un elemento (formando parte de este); es necesario utilizar la propiedad “content”

```
a::after { content: “ ¡Enlace! “ }
```

:focus afecta a los elementos que tienen el foco (por tanto, tan solo es aplicable a enlaces, botones o input, pues son los únicos elementos que pueden tener el foco)

:hover (para un elemento que tiene el puntero del ratón encima)

:active (para un elemento que está siendo pinchado)

:first-line permite aplicar estilos a la primera línea de un texto (según el medio cambiará el número de palabras que pertenezcan a esta primera línea)

```
p:first-line {text-transform: uppercase; }
```

:first-letter se utiliza para aplicar estilos a la primera letra de un texto, como se suele hacer al principio de los capítulos de un libro.

Ejemplo de primera letra modificada

Estilos que se suelen aplicar a la primera letra:

```
p:first-letter {

        font-size: 2.5em;
        font-weight: bold;
        line-height: .9em;
        float: left;
        margin: .1em;
}
```

:nth-of-type(n) selecciona los elementos que sean el hijo “n” de su padre (el valor de “n” puede ser un número, una fórmula o una palabra clave -*even*/*odd*); la fórmula será siempre a*n+b

```
tr:nth-of-type(even) { background-color: silver; } colorea las filas pares
tr:nth-of-type(3n+2) { background-color: silver; } selecciona cada tres filas, comenzando la secuencia en el segundo elemento
```

:first-of-type = :nth-of-type(1) selecciona el primer elemento de ese tipo para cada padre

:nth-last-of-type() como :nth-of-type pero empieza a contar por el último elemento

:last-of-type = :nth-last-of-type(1)

:nth-child(n) selecciona los elementos, sin importar el tipo, que sean el hijo “n” de su padre

```
/* Alternate paragraph colours in CSS */
        p:nth-child(4n+1) { color: navy; }
        p:nth-child(4n+2) { color: green; }
        p:nth-child(4n+3) { color: maroon; }
        p:nth-child(4n+4) { color: purple; }

    section tr:nth-child(3n+1) {
            background-color: grey;
    }
```

:nth-last-child()

:first-child = :nth-child(1) – ul li:first-child {…}

:last-child = :nth-last-child(1) – ul li:last-child {…}

COLORES

Los colores se pueden indicar de varias maneras: con palabras clave, colores del sistema, RGB hexadecimal, RGB numérico, RGB porcentual, etc.

1) Palabras clave

Hay muchas palabras clave que sustituyen al código de color. A continuación algunas de las más comunes.

black · white · brown · grey · silver · gold · pink · magenta · purple · plum · violet · salmon · red · maroon · crimson · coral · beige · cyan · aquamarine · blue · indigo · lavender · olive · green · lime · orange · tan · wheat · yellow

2) RGB

Define un color indicando la cantidad de rojo (R), verde (G) y azul (B) que lo forman. Hay varias formas de indicar estas cantidades.

2.1. *Decimal*: se indican tres valores entre 0 y 255 (se le da un peso a cada color).

color: rgb(0, 0, 0); // sería el negro

2.2. *Porcentual*: se indican los porcentajes de cada color.

color: rgb(27%, 38%, 69%);

color: rgba(100%, 0%, 0%, 0.5);

2.3. *Hexadecimal*: se indica el valor hexadecimal correspondiente a cada color. Cuando los dos valores de un color son iguales se puede resumir el código así: #33FF99 = #3F9.

color: #4762B0; // 47 (rojo), 62 (verde), B0 (azul)

Hay una versión del modelo RGB que añade la propiedad “alpha” de opacidad/transparencia (RGBA). Esto también ocurre con el siguiente modelo que veremos: el HSL o HSLA. Esta propiedad existe también por separado (**opacity**: 0.2; siendo 0 transparente y 1 sin transparencia).

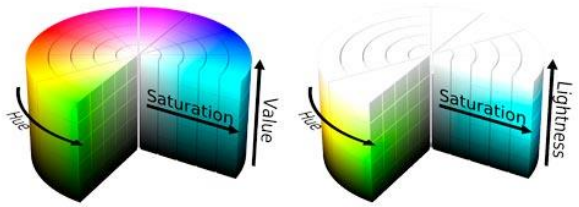
3) HSL

Las siglas HSL (hue, saturation, lightness) representan el matiz o frecuencia dominante del color dentro del espectro visible (de 0 a 360: 0 es rojo, 120 verde y 240 azul), la saturación o intensidad del color (en porcentaje) y la luminosidad o cantidad de luz (en porcentaje). Este sistema recorre los 360º del círculo cromático, de ahí el rango 0-360.

```
color: hsla(0, 100%, 50%, 0.5);
```

Usar HSL hace más sencillo crear armonías de colores variando alguno de los tres valores: paletas monocromáticas, paletas de colores análogos, complementarios, triadas, etc.

En el Color Picker nos encontraremos con las casillas H, S, B, que representan el sistema Hue Saturation Brightness/Value, que no es exactamente lo mismo que HSL pero es parecido: luminosidad es algo así como "cantidad de blanco" y brillo es más bien "cantidad de luz". A continuación la explicación gráfica y las ecuaciones de conversión HSL/HSB.



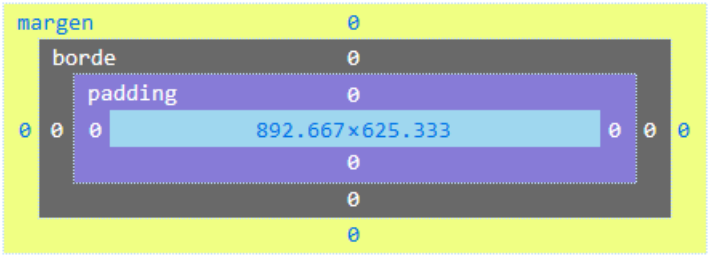
$$L = \frac{1}{2} B(2 - S_{HSB})$$

$$S_{HSL} = \frac{BS_{HSB}}{1 - |2L - 1|}$$

4) Colores del sistema: se pueden utilizar los colores del sistema operativo del usuario mediante unos códigos que se refieren a estos.

5) Web safe: este sistema consta de 216 de los 256 colores que veían los monitores de los '90.

MODELO DE CAJAS



Caja del elemento visible en el inspector de elementos del navegador

Todos los elementos de las páginas web se representan mediante cajas rectangulares que se crean automáticamente alrededor de ellos. Estas cajas están formadas por seis partes (de mayor a menor visibilidad, desde la que está al frente hasta la del fondo):

· **Contenido** (*content*): es el contenido HTML del elemento.

· **Relleno** (*padding*): espacio opcional que hace de marco entre el contenido y el borde. En esta zona se verá el color o imagen de fondo del elemento.

· **Borde** (*border*): línea que encierra el contenido y su relleno.

· **Imagen de fondo** (background image): imagen que está tras el contenido y el relleno.

· **Color de fondo** (*background color*): color que se muestra tras el contenido, el relleno y la imagen de fondo.

· **Margen** (*margin*): separación opcional entre la caja del elemento y las cajas adyacentes. En este espacio se verá el color o imagen de fondo del elemento padre (o de la página si este no está definido).

El relleno y el margen son transparentes.

Podemos ponerle al elemento estas partes (dándoles tamaño) o no (con lo cual la propiedad seguirá ahí latente, pero si no la incluimos en el estilo no se aplicará sobre el elemento -es como si no estuviera).

Para saber lo que ocupa en total mi elemento tengo que sumar las dimensiones de cada parte: 2 x margen + 2 x borde + 2 x padding + 1 x contenido

En el inspector del navegador (Firefox: F12 >> Calculado) posando el cursor sobre los recuadros de colores (el modelo de caja del inspector) veré en la ventana principal del navegador dónde están cada una de las partes del elemento seleccionado.

PROPIEDADES DE LA CAJA DEL ELEMENTO

· **width** (anchura del elemento), min-width, max-width (aplicables a todos los elementos menos filas de tablas). La propiedad "width" solo afecta al contenido (si le pones bordes y márgenes y tal, el tamaño de estos se sumará a la anchura del contenido). No se suele tocar más que cuando estamos modificando el *layout*.

  · **max-width** (um/none): anchura máxima de un elemento

  · **min-width** (um): anchura mínima de un elemento

· **height** (altura del elemento), min-height, max-height (son aplicables a todos los elementos menos columnas de tablas). No deberíamos tocar la altura (en principio) nunca porque el navegador la modifica para que se vea bien el elemento según el dispositivo en el que se esté proyectando. Si limitamos la altura podemos provocar *overlapping*.

· **max-height** (um/none): altura máxima de un elemento

· **min-height** (um): altura mínima de un elemento

· **background** (del contenido y el padding): aúna las propiedades *background-color* (color/transparent) y *background-image* (url/none). La imagen aparece delante del color de fondo (pero se suele indicar este también), y si es más pequeña que el fondo se repite formando un mosaico hasta rellenarlo; dentro de la propiedad imagen se pueden indicar las siguientes propiedades:

background-repeat (repeat/no-repeat/repeat-x/repeat-y) que controla si se repite y cómo lo hace (si horizontal o verticalmente)

background-position: medida respecto de la esquina superior izquierda (donde se coloca la imagen por defecto), si lleva dos valores es el desplazamiento horizontal y el vertical; también puede indicarse mediante palabras clave (left/center/right y top/center/bottom)

background-attachment (scroll/fixed -para que el fondo haga scroll o no se desplace con nosotros).

Ejemplo de múltiples fondos

```
{ background: #CCC url(…) right bottom no-repeat, url(…) left top repeat-x; }
```

o lo que es lo mismo:

```
{
        background-image: url(…), url(…);
        background-repeat: no-repeat, repeat-x;
        background-color: #CCC;
        background position: right bottom, left top;
}
```

· **margin** (tamaño del margen), como propiedad *short-hand* admite hasta cuatro valores (si tiene uno será el aplicable a los cuatro márgenes, si son dos serán superior/inferior y derecho/izquierdo; si son tres serán superior, derecho/izquierdo e inferior; si son cuatro serán superior, derecho, inferior, izquierdo) que sustituyen a *margin-top*, *margin-right*, *margin-bottom* y *margin-left*. Los márgenes superior e inferior solo se aplican a elementos de bloque e imágenes, y se solapan con los márgenes inferior y superior de las cajas de los elementos adyacentes (en caso de haber dos márgenes será visible el que sea mayor). En la zona de margen el color es transparente: no se puede definir color (lo que vemos es el color de fondo del elemento contenedor).

Si quiero que todos sean del mismo tamaño menos uno:

```
p {
margin: 30px;
margin-right: 10;
}
```

Al poner esto: { margin: 10px 20px 30px 40px; } estaré diciéndole: margin-top: 10px; margin-right: 20px; margin-bottom: 30px; margin-left: 40px;

· **padding** (relleno del elemento): puede utilizarse igual que margin como propiedad resumen de *padding-top*, *padding-right*, *padding-bottom*, *padding-left*. Los padding no se solapan.

· **border** (aspecto del borde: anchura, color y estilo): se pueden hacer todas las combinaciones posibles entre la propiedad (width, color, style) y el lado del borde (superior, derecho, etc.), como border-top-width, border-right-color, border-bottom-style, etc. Si hay algo que tiene borde y lo queremos quitar: { border: none; }

border-width (palabras clave: thin, medium, thick)

border-color (color / transparent)

border-style (none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset) – none y hidden muestran lo mismo, pero no se comportan igual al entrar en conflicto con bordes de celdas adyacentes en tablas

Propiedad resumen: border: 1px solid #369;

{ border-style: double dotted; } arriba y abajo doble, izq y dcha dotted

La anchura y altura total de los elementos es la suma de su width/height y sus márgenes, padding y bordes.

Es interesante tener en cuenta que para elementos de bloque se puede utilizar un borde inferior como si fuera subrayado: h1 { border-bottom: 2px solid blue; }

<u>border-image</u> (pone una imagen como borde): Si solo pongo el enlace a la imagen y la anchura lo que hace es ponerme la imagen completa en cada esquina del borde. Se puede utilizar una imagen troceada para que las distintas partes del marco tengan imágenes distintas (hay que indicarle cómo se trocea la imagen y cómo se aplica).

```
border-image-slice: 33% 33% 33% 33%; // cortes de la imagen
border-image-width: 1.5em 1.5em 1.5em 1.5em;
border-image-outset: 0px 0px 0px 0px;
border-image-repeat: round round; // vs stretch vs repeat
border-image-source: url("https://mdn.mozillademos.org/files/6009/border-image-2.png");
```

<u>border-radius</u> (aplica bordes redondeados a las esquinas), se indica su radio (px); si se indican 2 medidas la primera es para la esquina superior izquierda y la inferior derecha; si se indican 3 medidas, la primera es para la izq superior, la segunda para la dcha superior y la izq inferior, y la tercera para la inferior dcha.

· **box-shadow** (none/um): le añade sombra a la fuente; los valores que se le pueden dar son horizontal(px) vertical(px) blur(px) color inset spread (h, v y color son obligatorios); si quiero poner la sombra hacia el lado contrario tengo que poner números negativos. Ej. 40px 30px 10px #444

· **box-sizing**: border-box; (hace que se cuenten el borde, padding y margen dentro de la anchura y altura indicadas); se suele aplicar a todos los elementos: * { box-sizing: border-box; }

POSICIONAMIENTO

Los navegadores tienen en cuenta varios factores para crear la caja de cada elemento: su anchura, altura, que sea en bloque o en línea, el posicionamiento de la caja (normal, relativo, absoluto, fijo o flotante), las relaciones entre elementos y otros (tamaño de las imágenes, medio, etc.).

Podemos darle propiedad posicionamiento relativo u absoluto a un contenedor con el objetivo de que sus elementos se muevan con respecto a este posicionamiento.

· **position** (static/relative/absolute/fixed): indica la posición de la caja.

El posicionamiento normal (*static*) es el que usan los navegadores si no se indica lo contrario. Para él se tiene en cuenta el tipo de elemento (bloque/línea), su altura y anchura. Los elementos de bloque forman en CSS <<contextos de formato de bloque>> (las cajas se muestran una debajo de otra); si el elemento se encuentra dentro de otro, el contenedor o padre determina la posición y el tamaño de las cajas interiores -el padre supremo es el *body*. Los elementos en línea forman <<contextos de formato en línea>> (las cajas se muestran una detrás de otra de forma horizontal).

En el posicionamiento relativo (*relative*) la caja se desplaza respecto de su posición original (la que tendría de tener posicionamiento estático). El desplazamiento se controla con las propiedades *top*, *right*, *bottom* y *left* (que desplazan la caja con respecto al borde mencionado, es decir, 15px top serían 15px por debajo del borde superior "estático" y -15px right sería igual que 15px left). El desplazamiento de una caja no afecta a las adyacentes (se respeta el espacio que ocupaba el elemento desplazado). Por defecto, el desplazamiento está fijado a 0,0; así, podemos posicionar un elemento (como relativo) sin moverlo si no indicamos valores.

```
position: relative;
top: 200px;
left: 200px;
```

En el posicionamiento absoluto (absolute) la posición de la caja se establece de forma absoluta respecto del punto superior izquierdo de su primer elemento contenedor que tenga un posicionamiento distinto al estático/normal (o, si no lo hay, respecto al body); afecta a otros elementos (el hueco no se respeta).

```
position: absolute;
top: 200px;
left: 200px;
```

El posicionamiento fijo (fixed) es posicionamiento absoluto e inamovible (al hacer el scroll).

· **float** (left/right/none): hace que un elemento se vuelva flotante

El posicionamiento flotante desplaza la caja todo lo posible hacia un lado de la línea en que se encuentra y la caja deja de pertenecer al flujo normal de la página, influye en la disposición de las demás cajas (los elementos adaptan su contenido para que fluya alrededor de esta caja). Se utiliza principalmente para crear *layouts* o para que un texto fluya alrededor de una imagen/tabla/otro párrafo (se flotará la imagen).

Si queremos que un elemento no fluya alrededor de un elemento flotante que le precede le añadiremos la propiedad clear: both;

· **clear** (none/left/right/both): evita que un elemento flote alrededor de otros

Si quiero hacer que un párrafo aparezca como un recuadro separado al lado del resto del texto:

```
{
        float: left;
        width: 20%;
        padding: 0.2em;
        border: 2px solid blue;
}
```

VISUALIZACIÓN

· **display** (inline/block/none/list-item/run-in/inline-block/table/inline-table/table-row-group/table-header-group/table-footer-group/table-row/table-column-group/table-column/table-cell/table-caption): permite ocultar completamente un elemento (los demás ocuparán su lugar), hacer que un elemento se comporte como si fuera de bloque y otras cosas.

· **visibility** (visible/hidden/collapse): *hidden* permite hacer invisible un elemento (crea la caja pero no la muestra, el hueco que ocuparía el elemento permanece vacío), los demás elementos no modifican su posición (la caja sigue ocupando su sitio aunque sea invisible); *collapse* oculta filas o columnas de una tabla.

· **overflow** (visible/hidden/scroll/auto): controla la forma en que se ven los contenidos que sobresalen de sus elementos (para elementos de bloque y celdas de tablas); al aplicarle *visible* el contenido se muestra aunque sobresalga, con *hidden* se oculta el contenido que no cabe en la zona reservada, con *scroll* aparece una barra de desplazamiento para ver el contenido que sobresaldría (la barra aparecerá aunque no haga falta) y con *auto* depende del navegador (normalmente pone un *scroll* si hace falta). El overflow solo se usa si estoy limitando la altura (height: 300px) o trabajando con elementos flotantes.

Al flotar el contenido de un elemento, este "desaparece" y es necesario darle overflow: auto/hidden; para que "ocupe su lugar" (en lo que a margen se refiere).

· **z-index** (auto/numero): controla en qué orden se muestran las cajas cuando hay solape (de adelante hacia atrás: a mayor número, más cerca del usuario está la caja; 0 se considera el menor; se suelen utilizar 10, 20, 30, etc., por si luego quiero intercalar cosas nuevas); solo tiene efecto en elementos posicionales.

TEXTO

· **color**: color de la letra

· **font-family** (nombre de fuente o familia): se suele poner una lista de tipos de letra en orden de preferencia. Además de los nombres de fuentes, hay una serie de palabras clave que engloban familias de fuentes: monospace, serif (con serifa o adorno), sans-serif (sin serifa). Listas típicas:

```
Verdana, Arial, Helvetica, sans-serif;

Georgia, “Times New Roman”, Times, serif;

“Courier New”, Courier, monospace;
```

Cuando se quiere mostrar una fuente especial se puede crear una imagen con el texto (para textos cortos) o sustituir automáticamente el texto con Flash (SIFR).

Otra opción es utilizar una fuente personalizada que subiremos desde un fichero local:

1. Asociar el fichero con el nombre que le daremos a la fuente de cara al CSS:

```
@font-face {
        font-family: mona;
        src: url(“fuentes/Mona Shark.otf”);
}
```

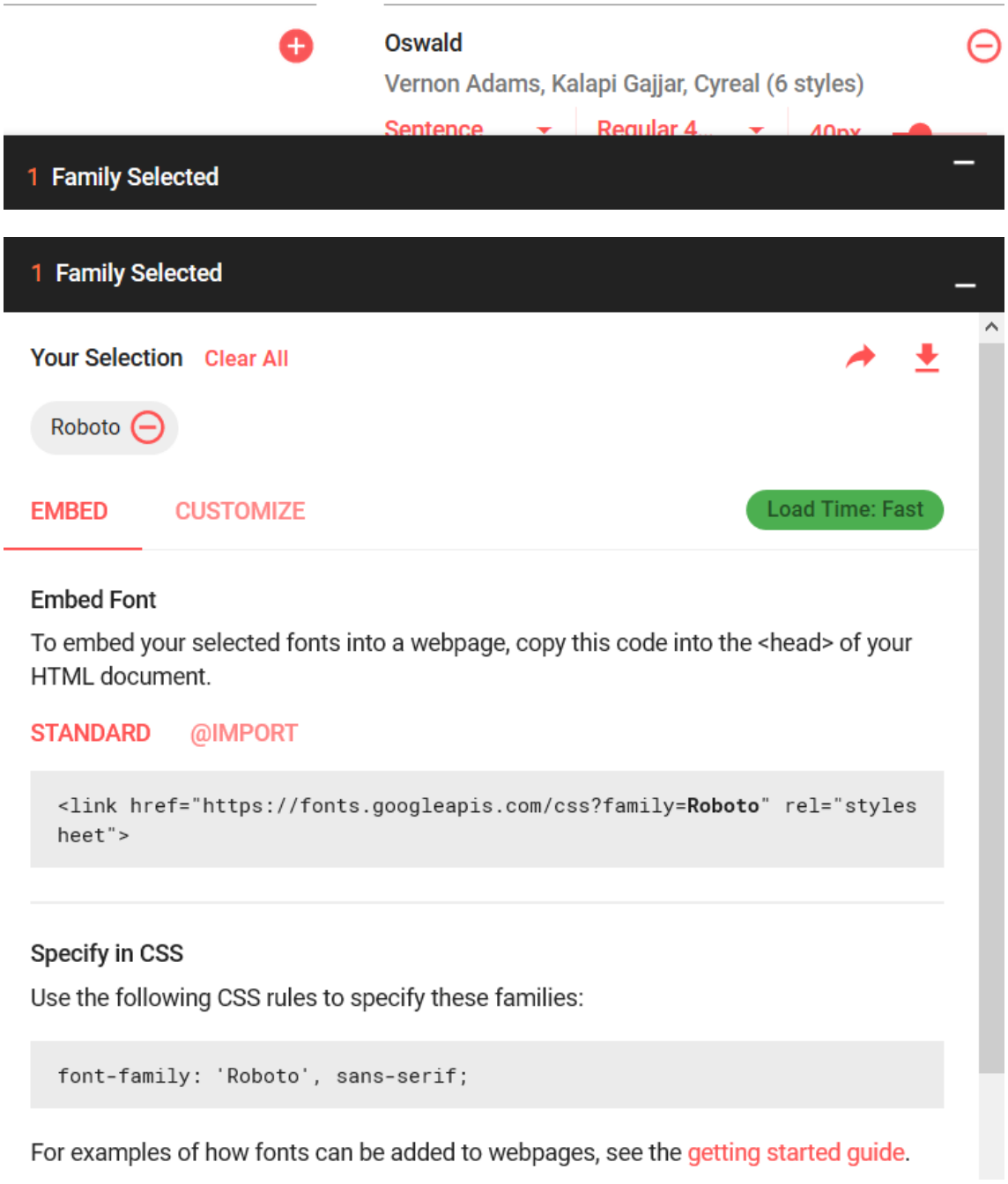
2. Utilizamos la fuente en el CSS llamándola por el nombre que le acabamos de dar:

```
#contenido {
        font-family: mona;
 }
```

Y también se puede utilizar las fuentes proporcionadas por Google (fonts.google.com) o *webfonts*. Para esto hace falta seleccionar la fuente deseada e incluir un <link rel=”stylesheet” href=”url”> con la url de la fuente y ya se podrá utilizar en el CSS con el nombre de la fuente. No es muy recomendable utilizarlas para navegación móvil, ya que si el navegador tarda mucho en recibir la fuente utilizará la que tenga por defecto. Pasos a seguir para utilizar una fuente de Google:

Voy a fonts.google.com. Cuando sé qué fuente quiero le doy al + de la derecha de su nombre y voy a la pestaña que dice “1 Family Selected” de abajo a la derecha del navegador: se desplegará y mostrará el enlace a la fuente que hay que agregar a la head de HTML, así como la línea de CSS que indicará el uso de esta fuente.

fonts.google.com >> fuente elegida >> + >> family selected (abajo a la derecha) >> embed: standard




```
<head>
        <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Tangerine">
        <link rel="stylesheet" href="webfonts.css">
</head>
```

Código CSS para utilizar esta fuente externa:

```
.googlefonts{
        font-family: 'Tangerine', serif;
}
```

Y ahora debo ponerle la clase "googlefonts" a los elementos que quiero que utilicen esta fuente.

· **font-size** (tamaño absoluto o relativo): se recomienda ponerlo en *em* o *%* (pt para impresión); aunque también hay palabras clave (xx-large, x-large, large, medium, small, xx-small, smaller, larger).

· **font-weigth** (normal/bold/bolder/lighter/100/200/…/900) grosor de la letra (normal = 400, bold = 700)

· **font-style** (normal/italic/oblique): estilo de letra

(Itálica y cursiva no es lo mismo: la "itálica" no tiene por qué ser cursiva, cuando se define una fuente se decide cómo es la itálica y no tiene por qué ser "tumbada".)

· **font-variant** (normal/small-caps): para aplicar un estilo de letra versal (versales o versalitas, que son mayúsculas con tamaño de minúsculas)

Propiedad resumen: **font**, que engloba (y deberían indicarse en este orden) font-weight, font-style, font-variant (opcionales); font-size (obligatorio) y line-height (opcional); font-family (obligatorio).

En lugar de indicar todos estos parámetros, también se puede definir la fuente a utilizar mediante una serie de palabras clave (caption/icon/menu/message-box/status-bar/small-caption) que indican la fuente utilizada por el sistema operativo para diferentes elementos (status-bar: barra de estado de las ventanas, icon: iconos, etc.).

· **text-align** (left/right/center/justify): alineación del contenido del elemento.

· **line-height** (normal/unidad de medida): altura de línea del elemento (múltiplo del tamaño de letra) o interlineado. { line-height: 1.2 | 120% }

· **text-decoration** (none/underline/overline/line-through/blink): decoración del texto (subrayado, con línea por encima, tachado, parpadeante). Es interesante recordar que con la propiedad border-bottom se puede aparentar un subrayado diferente (de distinto color que el texto, más largo, etc.).

· **text-shadow** (horizontal vertical blur color) Ej. p { text-shadow: 2px 2px 5px red; }

· **text-transform** (none/capitalize/uppercase/lowercase): transforma el texto a mayúsculas, minúsculas o pone la primera letra de cada palabra en mayúscula (*capitalize*).

· **vertical-align** (baseline/sub/super/top/text-top/middle/bottom/text-bottom): alineación vertical de los contenidos de un elemento (para elementos en línea y celdas de tabla).

· **text-indent** (unidad de medida) aplica una sangría a la primera línea de cada párrafo.

· **letter-spacing** (normal/u.m.): espacio entre las letras de cada palabra (el valor puesto se añade a la separación por defecto)

· **word-spacing** (normal/u.m.): espacio entre las palabras

· **white-space** (normal/pre/nowrap/pre-wrap/pre-line): establece el tratamiento de los espacios en blanco

VALOR	RESPETA ESPACIOS EN BLANCO	RESPETA SALTOS DE LÍNEA	AJUSTA LAS LÍNEAS
normal	no	no	sí
pre	sí	sí	no
nowrap	no	no	no
pre-wrap	sí	sí	sí
pre-line	no	sí	sí

ENLACES

El estilo básico para los enlaces supone cambiar el tamaño, color de letra y decoración del texto. Por defecto aparecen en azul y subrayados (el navegador controla que el color y la anchura del subrayado sea acorde al tamaño de letra -una forma de controlar esto es utilizar la propiedad border-bottom).

Hay una serie de pseudo-selectores que se pueden aplicar a los enlaces:

:link (para enlaces que aun no han sido visitados)

:visited (para enlaces que ya han sido visitados)

:focus (para enlaces que tienen el foco)

:hover (para un enlace que tiene el puntero del ratón encima)

:active (para un enlace cuando está siendo pinchado)

Los dos últimos sirven para todos los elementos HTML.

Imágenes al lado de un enlace

Las añadiremos con CSS (imagen de fondo y padding).

Por ejemplo, puedo poner una **imagen a la izquierda de cada enlace que lleve a un pdf**: creo un padding para que no se solape, pongo una imagen de fondo q no se repita a la izquierda:

```
a[href$=".pdf"] {
padding-left: 20px;
background: url(imagenes/pdf.png) no-repeat left center;
}
```

LISTAS

· **list-style-type** (disc/circle/square/decimal/decimal-leading-zero/lower-roman/upper-roman/lower-greek/lower-latin/upper-latin/armenian/georgian/lower-alpha/upper-alpha/none): establece el tipo de viñeta mostrada para una lista; "none" se suele utilizar para menús de navegación.

· **list-style-position** (inside/outside): establece la posición de la viñeta (con sangría -aparece en la zona de padding- o alineada con el resto del párrafo -aparece en la zona de contenido).

· **list-style-image** (url/none): reemplaza la viñeta por una imagen; (la imagen debería ser más pequeña que el tamaño de la letra). list-style-image: url(imagen.jpg);

· **list-style** (propiedad resumen) – Ejemplo: ul { list-style: url("…") square inside; }

Cómo hacer un menú vertical

```
nav ul  {
        list-style-type: none;
        margin: 0;
        padding: 0;
        width: 200px;
}
nav ul li {
        background: #F4F4F4;
        border: 1px solid #7C7C7C;
}
nav ul li a {
        display: block;
        text-decoration: none;
}
```

Cómo hacer un menú horizontal

```
nav ul  {
        background:  #F4F4F4;
        border: 1px solid #7C7C7C;
        box-sizing: border-box;
        clear: both;
        list-style-type: none;
        margin: 0;
        padding: 0;
        overflow: hidden;
}
nav ul li {
        float: left;
        width: 20%;
}
nav ul li a {
        border-left: 1px solid #FFF;
        border-right: 1px solid #FFF;
        color: #333;
        display: block;
        padding: .3em;
        text-decoration: none;
}
```

CONTADORES

Las propiedades que utilizaremos para crear un contador (una serie de números que se van incrementando automáticamente y aparecen al lado de los elementos que indiquemos) son:

· **counter-reset** (crea un contador por primera vez o lo resetea al punto inicial): hay que indicarle el nombre del contador y el valor que se le da al inicio (por defecto empezará a numerar por el 1).
· **counter-increment** (incrementa el valor del contador)
· **counter()** (añade el valor de un contador a un elemento)

1. Para introducir un contador en ciertas partes del documento, primero hay que inicializarlo (darle un nombre y un punto de inicio).

```
counter-reset: nombre -1; (el "-1" sería para que empiece a numerar por el 0)
```

Hay que tener cuidado con esta propiedad, pues en caso de inicializar dos contadores dentro del mismo selector solo hará efecto el segundo, ya que CSS lo entenderá como una colisión.

2. Después, hay que indicar el incremento que queremos que se le sume en cada ocurrencia (es necesario indicar el nombre del contador porque es común trabajar con varios a la vez):

```
counter-increment: nombre 2; (para que el contador se incremente de 2 en 2)
```

3. Y decirle que lo muestre con el formato deseado donde queramos que aparezca:

```
content: counter(nombre, ".") "-" ; (los signos son embellecedores comunes pero innecesarios)
```

```
dl{
counter-reset: ni;
}

dt::before {
        counter-increment: ni;
        content: "(" counter(ni) ") - " ;
}
```

Se puede poner un contador con un formato que no sea numérico (por ejemplo con letras). Esto se debe especificar a la hora de introducir el dato que se va a mostrar.

```
content: counter(nombre, lower-alpha);
```

Para mostrar dos contadores juntos en el mismo espacio (por ejemplo para numerar las casillas de una tabla) hay que hacerlo así:

```
content: counter(numerosajedrez) counter(letrasajedrez, lower-alpha);
```

Para introducir un contador en la celda de una tabla es necesario utilizar el selector td:before o td:after, por alguna razón no muestra nada si se pone simplemente "td".

Para numerar listas anidadas hay que quitar las viñetas primero:

```
ol {
        counter-reset: nlista;
        list-style-type: none;
}

li::before {
        counter-increment: nlista;
        content: counter(nlista) + " ";
}
```

Ejemplo – para que me numere los h2, h3 que están dentro de un artículo, inicializo el contador con el selector de artículos y reinicializo en cada h2:

```
article { counter-reset: nh2; }
article h2{ counter-reset: nh3; }

article h2::before {
        counter-increment: nh2; // incrementa en 1 el contador "nh2"
        content: counter(nh2) "."; // muestra el contador autoincrementado en cada
h2
}
article h3::before {
        counter-increment: nh3;
        content: counter(nh2) "." counter(nh3) ".";
}
```

TABLAS

· **border-collapse** (collapse/separate): fusión de bordes de celdas adyacentes o no (borde doble).

(1) border-collapse: collapse;

(2) border-collapse: separate; esta opción libera otras dos propiedades aplicables:

border-spacing (um): separación entre bordes de celdas adyacentes (horizontal vertical)

empty-cells (show/hide): muestra u oculta las celdas vacías

· **caption-side** (top/bottom): posiciona el título de la tabla encima o debajo de esta.

De esta forma (y con caption-side: bottom) puedo poner la descripción debajo a la izquierda:
caption{ text-align: left; }

Para las celdas que están fusionadas con otras:

td[colspan], td[rowspan] {}

Para aplicarle una propiedad (color de fondo por ejemplo) a las filas pares o impares de una tabla:

tr:nth-child(even) {background: #ddd;} // even (par) = 2n

tr:nth-child(odd) {background: #fff;} // odd (impar) = 2n+1

FORMULARIOS

Ejemplos útiles:

```
input { padding: .2em;  }

label { display: block; margin: .5em 0 0 0;  }
```

:focus (pseudo-elemento que hará efecto cuando el elemento en cuestión tenga el foco -por tanto, tan solo es aplicable a enlaces, botones o input pues son los únicos elementos que pueden tener el foco)

```
input:focus { background-color: yellow; }
```

Para poner un formulario en dos columnas (que en lugar de aparecer los apartados uno debajo de otro aparezcan uno al lado del otro) podemos bien poner el texto en columnas o hacer que floten los “fieldset”:

```
fieldset {
        box-sizing: border-box;
        float: left;
        margin: 0;
        width: 50%;
}
```

LAYOUTS

Para centrar una página horizontalmente lo mejor es guardar todo el contenido de la página en un < > (id=”contenedor”) y aplicarle { margin: auto; } y una anchura (si queremos) del, por ejemplo, 70%.

Si hay algo debajo de los elementos flotantes que no queremos que se mueva hay que aplicarle { clear: both; } .

Cuando una sección tiene todos sus elementos flotando, pueden pasar cosas raras: el color de fondo solo aparece en la cabecera pues los demás elementos están flotando. Para evitar esto le digo "section{overflow: auto;}"

Es importante recordar también añadir la propiedad *box-sizing: border-box;* para no tener problemas con la anchura de los elementos.

Así se divide en dos columnas y aparece a la izquierda el header y a la derecha la sección:

```
header, section {
        float: left;
        width: 50%;
}

Así aparecería la cabecera a la derecha:
section {
        float: left;
        width: 75%;
}
header {
        float: right;
        width: 25%;
}
```

Si quiero poner un footer con (por ejemplo) flechas de página siguiente y página anterior le puedo poner clase "izq" a una y clase "dcha" a otra y poner en el css:

```
.dcha{
        float: right;
}
.izq{
        float: left;
}
```

Si ahora le pongo un borde al footer solo me aparecerá el borde superior. Esto ocurre porque el footer tiene una altura 0 ya que al flotar sus imágenes se salen del contenedor.

```
footer {
border: 2px solid blue;
overflow: auto; | hidden; // así el marco recuadra las imágenes
}
```

Float no es la única forma de hacer layout. También se pueden hacer con table-cell, inline-blocks, flex (no está orientado a hacer layout, pero se puede utilizar para eso) y grid.

· **columns**: La propiedad *column* reparte el texto en varias columnas, no se puede elegir qué parte del texto aparece en cada columna (se irá moviendo de una a otra según el tamaño de la ventana). Es una propiedad resumen de *column-count* y *column-width*.

Aparte de estas, también existen las propiedades *column-gap* (espacio entre columnas), *column-span: all* (para que se expanda el título por todas las columnas), *column-rule* (que tiene los mismos valores que el borde y es resumen de *column-rule-color*, *column-rule-style*, *column-rule-width*).

Existe, por ejemplo, una técnica llamada *faux columns* que consiste en simular columnas poniendo una imagen de fondo del elemento principal contenedor con las columnas dibujadas.

IMPRESIÓN

Lo más común es crear una hoja de estilos para impresión que se enlaza en el HTML con el atributo media="print" (la hoja de estilos básicos y las complementarias llevarán media="screen" u otros), pero si la página es sencilla y necesita pocos cambios puede ser más conveniente utilizar el contenido de la hoja de estilos básica y desactivar algunos elementos.

Si le doy a imprimir a la web me sale una previsualización de lo que se va a imprimir: si quiero que se imprima algo diferente de lo que se ve en pantalla (por ejemplo, que salga en blanco y negro o quitar el color de fondo, tamaño de letra: 12 puntos en lugar de los píxeles que dependen de la pantalla).

Cosas que se suelen poner en el diseño de impresión para eliminar contenidos:

```
#cabecera, #menu, #comentarios, nav { display: none; }

body, #contenido, #principal, #pie { float: none; width: auto; margin: 0; padding: 0; }

body { color: #000; font: …; }

#contenedor { font-size: 14pt;  font-family: consolas, monospace; }
```

Para modificar los enlaces y que aparezca la dirección de destino tras la expresión enlazada utilizo la propiedad "content" junto con la función "attr" (que permite mostrar el valor de un atributo). Esto es interesante para que aun en el documento impreso se vea a dónde llevan los enlaces.

```
a:after { content: "(" attr(href) ")" ; }
```

(Si lo quiero poner bonito: " [url: " attr(href) "]" -todo lo que va entre comillas sale tal cual.)

DISEÑO ADAPTATIVO

El objetivo del diseño adaptativo es adaptar los contenidos al dispositivo que utilice en cada momento el usuario (móvil, tablet, ordenador, etc).

Aunque hay dos enfoques clásicos (mejora progresiva -de móvil a escritorio- y degradación progresiva -de escritorio a móvil-) hoy en día se diseña principalmente pensando en el móvil (*mobile first design*), y luego se hacen adaptaciones al resto de dispositivos.

En el diseño para móvil es común utilizar el diseño apilado o *stacked*, en el que el ancho de todos los elemento es el 100%.

Lo primero que hay que hacer cuando se va a realizar un diseño adaptativo es añadir el siguiente elemento en la cabecera del HTML para que el contenido se adapte a la anchura de la pantalla del dispositivo: <meta name="viewport" content="width-device-width">

Formas de incluir el diseño adaptativo en nuestro código

Para incluir el diseño adaptativo en diseño de la página se puede:

· Enlazar el CSS en la cabecera del HTML mediante el elemento *link*. Esto es lo más común.

```
<link rel="stylesheet" href="layout.css" media=”screen”>
<link rel="stylesheet" href="print.css" media="print">
```

· Importar una hoja de estilos desde otra con **import**. Esta regla debe estar al principio del documento CSS (solo por detrás de declaraciones @charset) y puede incluir media queries:

```
@import url("estilos-impresora.css") print;
@import url(“estilos-movil.css”) screen and (max-width: 768px);
```

· Introducir en el fichero CSS directamente las instrucciones (que preceden a las reglas que se aplicarán a cada medio)

Media queries

Las *media queries* son una técnica de CSS3 que utiliza la regla **@media** para incluir un bloque de propiedades CSS solo si una condición concreta es válida. Las condiciones que más se utilizan son el medio de acceso y el tamaño de la pantalla/navegador.

Para esto último existen una serie de puntos de ruptura (medidas mínimas y máximas -en píxeles- del navegador) que sirven para acotar el comportamiento de la página en función del navegador.

```
@media print {
        body { font-size: 10pt;
        }
}

@media (max-width: 560px) {
        aside {
                display: none;
        }
        body {
                background-color: green;
        }
}
```

También se puede combinar el medio de acceso con las restricciones de tamaño:

```
@media screen and (min-width: 560px) and (max-width: 960px) { … }
```

Y se puede indicar la orientación de la pantalla (vertical -*portrait*- u horizontal -*landscape*-):

```
@media screen and (orientation: landscape) { … }
```

Si no se incluye el tipo de medio se entenderá que es aplicable a todos.

Se pueden realizar combinaciones con los operadores AND (and), OR (,) y NOT:

```
@media all and (max-width: 699px) and (min-width: 520px), (min-width: 1151px) {
 body {
   background: #ccc;
 }
}
```

(esto quiere decir: entre 520 y 699, y a partir de 1151)

Diseños adaptativos comunes

Hay que recordar no mantener fijas las dimensiones de una imagen grande para que al visualizar la página con un dispositivo pequeño no haya que hacer scroll horizontal.

Barra de navegación horizontal para pantallas pequeñas

```
nav ul li {
        float: left;
        width: 25%    /* para una lista de 4 elementos */
}

nav ul {
        list-style-type: none;
}
```

OTROS

La forma de **comentar** el código en CSS es la siguiente: /* comentario */ . En CSS no se pueden anidar comentarios.

Hay propiedades que se **heredan**, o sea que si le adjudico una propiedad a un elemento, sus elementos hijos heredan esa propiedad. Normalmente las propiedades de texto se heredan (formato, color) y otras como el borde no. Si queremos forzar a que un elemento herede una propiedad de su padre/contenedor podemos utilizar el valor "inherit". Al heredar una propiedad numérica, se hereda el valor ya calculado (en el caso de que la medida sea relativa); es decir, si el valor original era 12px de tamaño de fuente y 3em de sangría (text-indent), el heredado será 12x3px, y no 3 x el tamaño de la fuente del elemento actual.

Las propiedades ***shorthand*** son propiedades resumen que agrupan múltiples propiedades y permiten ver los valores de todas ellas de una vez. (Por ejemplo, *margin*, *padding* y *border*.)

```
{ border-bottom: 2px solid blue; } agrupa las siguientes:
        border-bottom-width: 2px;
        border-bottom-style: solid;
        border-bottom-color: blue;
```

Vendor prefixes (prefijos de vendedor): Sirven para que las propiedades funcionen en navegadores que están probando propiedades nuevas y no las aceptan en su versión definitiva todavía.

```
border-image + tab =
-webkit-border-image: ;
-o-border-image: ; // Opera
border-image: ;
```

Para no tener que preocuparnos por esto, el Autoprefixer pone automáticamente los prefijos necesarios según la compatibilidad que le pidamos.

Otras propiedades:

· cursor (default/hand/pointer/move/wait/…/url): personaliza el puntero del ratón.

:link, :visited { cursor: url(…), url(…), pointer; }

BIBLIOGRAFÍA

· Libros Web (http://librosweb.es/libro/css/)

· W3 (www.w3.org/Style/CSS/current_work): En la web de www.w3.org está la información oficial sobre todo lo estandarizado hasta ahora. Una de las páginas más útiles para esto son:

CSS Snapshot 2017 (foto de cómo está CSS en cada momento)

Cascading Style Sheet level 2 revision 1 (CSS 2.1) specification

· W3 schools (http://www.w3schools.com/css/)

· Estándar actual de selectores: http://www.w3.org/TR/css3-selectors/

· Tabla resumen de selectores: http://www.w3schools.com/cssref/css_selectors.asp

· Para validar el código (comprobar que no tiene errores) existe el validador del W3C (*World Wide Web Consortium*) : http://jigsaw.w3.org/css-validator/

· Hay una página web que te dice si una propiedad concreta está soportada por un navegador concreto (si está trabajando con lo último: lo que está en el working draft).

· Colores: palabras clave (https://en.wikipedia.org/wiki/X11_color_names)

· Colores del sistema: http://www.w3.org/TR/CSS21/ui.html#system-colors

· Colores: ejemplos de paletas (http://www.colorhexa.com/)

· Colores: http://htmlcolorcodes.com/color-picker/

· Colores: hslpicker.com

· Colores: colorcodes.com

· Colores: paletton.com te da una serie de colores similares para utilizar en una página para que quede armoniosa

· https://websitesetup.org/web-safe-fonts-html-css/

· www.w3schools.com/cssref/tryit.asp

· Border-image generator de Mozilla: (aquí) + explicación del borde de imágenes (Explained)

(https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Background_and_Borders/Border-image_generator)

· https://caniuse.com/ (te dice qué navegadores soportan una propiedad concreta)