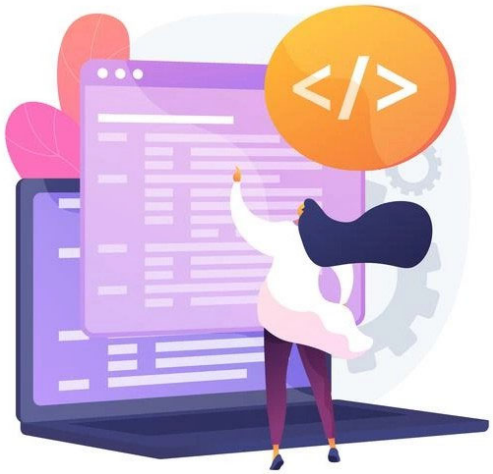


MF0491_3
PROGRAMACIÓN WEB EN EL ENTORNO CLIENTE

UF1841: elaboración de documentos web mediante lenguajes de marcas

El diseño web cubre la funcionalidad, la utilización y la estética

UF1841: ELABORACIÓN DE DOCUMENTOS WEB MEDIANTE LENGUAJES DE MARCAS

El diseño web se puede definir como tarea destinada a la planificación, el diseño y la implementación de páginas y sitios web. Es importante tener en cuenta:

- Navegabilidad.
- Interactividad.
- Usabilidad de la página.
- Arquitectura y distribución de la información presentada.

Diseño web.

Principios de diseño web.

El diseño web es una técnica basada en un conjunto de reglas más o menos definidas que sirven para crear páginas que sean útiles (usabilidad) y visualmente atractivas (aspecto).

Diseño orientado al usuario.

El diseño web centrado en el usuario se caracteriza por tener muy en cuenta las necesidades, características y objetivos que este desea alcanzar. Este tipo de diseño es necesario evaluar el sitio con los propios usuarios, analizar sus impresiones ante el diseño.

Diseño orientado a objetivos.

Definir los objetivos que se deseen alcanzar, se han de identificar los requerimientos del proyecto en cuestión. ¿Qué es lo que se pretende conseguir?.

Diseño orientado a la implementación

Consiste en centrar el diseño de la página en las posibilidades tecnológicas que están a la disposición del diseñador.

El proceso de diseño web.

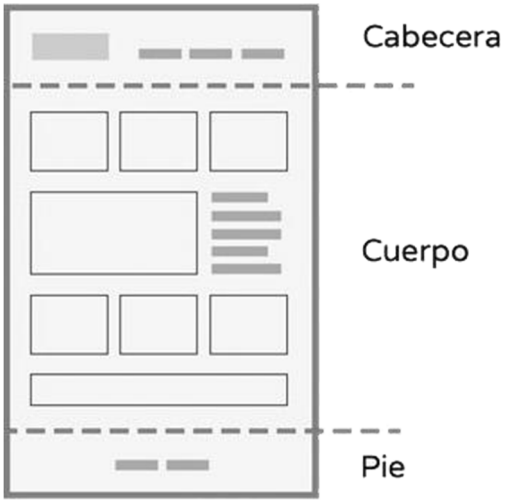
Estructura de un sitio web y navegabilidad.

Se trata de definir la manera en la que se enlazan las distintas páginas. Las ayudas para la navegación. Los hipervínculos entre los elementos.

Estructura y composición de páginas.

Las tres partes fundamentales en las que se suelen estructurar la inmensa mayoría de los documentos web es:

- La cabecera, con información identificativa del sitio, y aparece en la parte superior de la página.
- El cuerpo contiene lo más relevante de un documento web.
- El pie de página con información de contacto, menús que direccionan a las secciones más relevantes, links a otras páginas externas y, en general, contenido complementario relacionado con el tema de la página.



Compatibilidad con navegadores.

Es imprescindible verificar que se visualiza correctamente en, al menos, los navegadores más utilizados (firefox, chrome, safari) , ya que el HTML5 y CSS3 presenta comportamientos distintos según del navegador utilizado.

También se debe verificar que no existen links rotos, que no hay faltas de ortografía. Es aconsejable recabar la opinión de terceros sobre la funcionalidad de nuestra web.

Lenguajes de marcado generales.

Origen de los lenguajes de marcado generales.

Los lenguajes de marcas definen la estructura, la semántica y controlan el procesamiento de un documento digital.

SGML. Creado a partir de GML de IBM (solución a la visualización e impresión de un documento sin depender del hardware), se busca un estándar. Standard Generalized Markup Language.

XML. Utilizado para intercambio de datos. eXtensible HyperText Markup Language.

HTML. Es el lenguaje de marcas utilizado para la elaboración de páginas web.

Características generales de los lenguajes de marcado.

Texto plano. Se puede hacerse con cualquier editor de texto.

Compacidad. Las etiquetas y el contenido están mezclados. Las etiquetas aportan información sobre la estructura.

Flexibilidad. En sus inicios se utilizaban para incluirlos en documentos de texto, actualmente destaca la creación de páginas web.

Estructura general de un documento con lenguaje de marcado.

Metadatos e instrucciones de proceso.

Los metadatos consisten en información complementara (datos de datos) que se puede incluir en un documento.

Las instrucciones de proceso indican como se debe interpretar el documento.

Codificación de caracteres. Caracteres especiales (escape).

La codificación es el método utilizado para convertir un carácter a un sistema de representación. UTF-8 es el sistema que garantiza la presencia de los caracteres del idioma español (ñ, acentos, etc).

Etiquetas o marcas.

Es una palabra incluida entre los símbolos "<" y ">" y suelen tener una etiqueta de apertura y otra de cierre. La de cierre incorpora el símbolo "/".

Elementos.

Los elementos están formados por las etiquetas y el contenido:

```
<etiqueta> contenido </etiqueta>
<nombre>José García</nombre>
<poblacion>Vigo</poblacion
```

Atributos.

Indican alguna propiedad del elemento, están compuestos por un nombre y un valor.

Comentarios.

Son útiles para hacer anotaciones o aclaraciones sobre el código. Los comentarios comienzan con <!-- y se cierran con -->.

Lenguajes de marcado para presentación de páginas web

HTML

W3C es una comunidad internacional que trabaja para crear estándares para la WWW.

La actual versión es la 5, que se empezó a desarrolar en 2004, aunque a estas alturas no todos los navegadores interpretan igual las etiquetas, existiendo aún algunas que no tienen ningúnn tipo de soporte.

Esta versión incorpora nuevos elementos de diseño con valor estructural y semántico (información extra que otorgan las etiquetas, permitiendo un mejor rastreo deol documento).

Como en todos los lenguajes de marca, las etiquetas van encerradas entre los símbolos de < y >, y cada etiqueta que se abre, se debe cerrar, aunque hay excepciones.

Estructura de un documento.

```
<! DOCTYPE html >
< html lang="es">
        < head >
                < title > Título de la web </ title >
                < meta charset="UTF-8" >
        </ head >
        < body >
                Hola mundo !!!!!
        </body>
</ html >
```

Elemento raiz.

El documento html se inicia indicándole a los navegadores de que tipo de documento se

```
<!DOCTYPE html>
```

trata:

El nodo superior siempre es la etiqueta <**html**> que deberá ir acompañada del atributo

lang, que identificará el idioma del documento.

```
<html lang="es">
</html>
```

De este elemento raiz dependen los elementos <**head**> y <**body**>.

Elementos de la cabecera.

Los elementos de la cabecera no serán visibles en el navegador. Aquí va la información que necesitan el navegador, el servidor web y los motores de búsqueda (metadatos).

Van enmarcadas entre las etiquetas :

```
<head>
</head>
```

En esta primera aproximación haremos referencia a 2 imprescindibles :

- Título de la página, visible en la pestaña del navegador, también será utilizado cuando el usuario agregue la página a favoritos.

```
<title> Título de la web </title>
```

- Etiquetas meta, Se utilizan para añadir información sobre la página.

```
<meta charset="UTF-8" >                    // Juego de caracteres
<meta name="keywords" content="html, css, js, php">
<meta name="description" content="Programación web">
```

Elementos del cuerpo del documento.

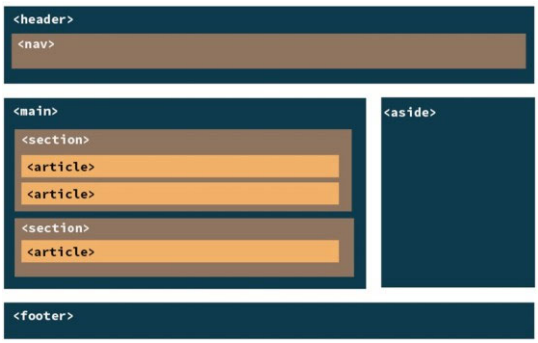
En el cuerpo del documento es donde se agregan los elementos visibles de nuestra web.

```
<body>
</body>
```

Estructura de un documento.

Hasta la llegada de HTML5 el principal elemento que se usaba para generar las estructuras era div.

Con la llegada de HTML5 nuestras webs utilizan otro tipo de contenedores, aunque se puede seguir utilizando div.



Esto se debe a la aparición de los nuevos contenedores semánticos, donde cada etiqueta indica el contenido que va tener.

Contenedores SEMÁNTICOS. Elementos de Cabecera y cuerpo del documento

<main>

Utilizada para el contenido principal, solo puede haber una en cada página. No puede utilizarse en el interior de otro elemento.

<header>

Está destinado a contener por lo general cabeceras, tanto de la web como de otros elementos. Es un elemento de *bloque*.

<footer>

En contraposición al header, este elemento es contenedor de pies de página o de otro elemento (artículo, noticias, post,…). Es un elemento de *bloque*, y al igual que el header, puede existir mád e uno en el documento HTML.

<nav>

Elementos de navegación con enlaces de hipertexto, pueden incluirse en un mismo

documento HTML todos los navegadores que sean necesarios.

Pero no todos los enlaces son candidatos de pertenecer a un elemento <**nav**>: una lista de enlaces a patrocinadores o los resultados de una búsqueda, no forman parte de la navegación principal, sino que corresponden con el contenido de la página. Es un elemento de *bloque*.

Es una práctica habitual anidar dentro de <**nav**> una lista desordenada con cada opción del menú

```
<nav>
        <ul>
            <li><a href="#" > Opción 1 </a></li>
            <li><a href="#" > Opción 2 </a></li>
            <li><a href="#" > Opción 2 </a></li>
       </ul>
</nav>
```

<section>

Etiqueta que permite agrupar elementos que comparten una misma temática y generalmente incluye una cabecera.

Puede usarse para dividir los documentos html u otros elementos en diferentes áreas o secciones.

Un **section** puede contener varios **article**. Es un elemento de bloque.

```
<section>
    <h2> Título de la sección </h2>
    <article>
        <p>
                Contenido del artículo: párrafos, imágenes, etc
        </p>
    </article>
</section>
```

<article>

Permite incluir contenido autónomo (que se pueda utilizarse en otros lugares del sitio), pudiendo anidar en su interior tanto cabeceras, pies u otros elementos.

Un **article** puede dividirse con **section** para separar su contenido. Es un elemento de *bloque*.

```
<article>
        <header>
            <h2> Título de la sección </h2>
        </header>
        <p>
            Contenido del artículo: párrafos, imágenes, etc
        </p>
        <footer>
                <p> pie del artículo </p>
        </footer>
</article>
```

<aside>

Representa una sección de una página con contenido tangencialmente relacionado con un elemento, y puede considerarse separado de este contenido.

Este elemento puede utilizarse para contener citas, anuncios, grupos de elementos de navegación y cualquier otro contenido separado del contenido principal de la pagina.

Es un elemento de *bloque*.

```
<aside>
// listas de elementos
// imágenes
</aside>
```

```
<aside>
        // listas de elementos
        // imágenes
</aside>
```


Se trata de un contenedor sin valor semántico y se puede utilizar en aquellos casos en los que se precise un bloque de diseño que no encaje en ninguno de los elementos anteriormente mencionados.

Es un *bloque* de diseño.

<hgroup>

La etiqueta <**hgroup**> representa el encabezado de una sección en grupo. Es decir, que nos permite encerrar títulos en bloques para que se tomen como un único elemento.

*El elemento <**hgroup**> se ha eliminado de la especificación HTML5 (W3C), pero todavía está en la versión WHATWG de HTML. Sin embargo, se implementa parcialmente en la mayoría de los navegadores, por lo que es poco probable que desaparezca (el validador www.validator.w3.org no lanza error cuando se incorpora esta etiqueta).*

Es un elemento de *bloque*.

```
<hgroup>
        <h1> Título principal </h1>
        <h2> Título secundario </h2>
</hgroup>
```

Color y codificación de colores

El color es un elemento fundamental en el diseño web, ya que influye significativamente en la percepción del usuario, la navegación, y la experiencia general. Su uso estratégico puede potenciar la identidad visual de una marca, evocar emociones específicas y guiar a los usuarios hacia acciones deseadas. Junto al color, las codificaciones, como los códigos de color en formato hexadecimal, RGB o HSL, son esenciales para la implementación técnica de los colores en el entorno digital.

Teoría del Color en el Diseño Web

Psicología del Color

Cada color puede evocar diferentes emociones y asociaciones culturales. Comprender esta psicología permite diseñar experiencias más efectivas y atractivas:

- **Colores cálidos (rojo, naranja, amarillo): Asociados con energía, urgencia y emoción. Por** ejemplo, el rojo suele usarse para botones de llamada a la acción.
- **Colores fríos (azul, verde, morado):** Asociados con calma, confianza y profesionalismo. El azul es popular en sitios corporativos y financieros.
- **Colores neutros (blanco, gris, negro):** Aportan balance, simplicidad y sofisticación.

Una tienda en línea de alimentos saludables puede usar verdes y tonos tierra para transmitir frescura y naturalidad.

Paletas de Color

Seleccionar una paleta de colores es clave para la coherencia visual de un sitio web. Una paleta suele incluir:

- **Color primario:** Representa la marca o el mensaje principal.
- **Colores secundarios:** Complementan al primario y añaden variedad.
- **Colores de acento:** Destacan elementos clave, como botones o enlaces.

En un sitio de educación en línea, el azul como color primario transmite confianza, mientras el naranja como color de acento destaca las llamadas a la acción.

Formatos de codificación del color

Los colores se representan en el diseño web mediante diferentes sistemas de codificación:

- **Hexadecimal**: el formato mas común en el diseño web. Representa los colores en un código alfanumérico de seis dígitos precedido por #

Ejemplo: #FF5733 representa un tono naranja

- **RGB (Red, Green, Blue):** Basado en la mezcla de colores primarios de luz. Se expresa con valores numéricos entre 0 y 255.

Ejemplo: rgb(255, 87, 51).

- **HSL (Hue, Saturation, Lightness):** Define los colores por matiz (hue), saturación y luminosidad.

Ejemplo: rgb(255, 87, 51)

Contenedores de TEXTO.

Además de los contenedores de bloques, existen contenedores especializados en textos, ya que es mediante el contenedor podemos dar formato a dicho texto.

Títulos - <h1> <h2> … <h6>

HTML5 proporciona 6 niveles de títulos, numerados del 1 al 6, que se corresponden con su importancia. Tienen también una configuración predeterminada con distintos tamaños.

<h1> es el de mayor nivel, y solo debería aparecer 1 en cada archivo html, que será el título principal.

No se recomienda abusar de estos elementos dentro de una página (no más de 3 niveles). Estos elementos tienen etiqueta de apertura y etiqueta de cierre.

```
<h1> Título principal </h1>
<h6> Título de menor valor </h6>
```

Párrafo -

La división de los textos en párrafos hace más comprensible la lectura y comprensión. Los párrafos son elementos de *bloque*, por lo que al cerrarse provoca un salto de línea (o bloque).

```
<p>Contenido del párrafo </p>
```

Listas

Las listas nos permite crear conjuntos de elementos en forma de lista dentro de una página, todos los cuales irán precedidos, generalmente, por un guión o número.

HTML incorpora 3 tipos de listas:

- Listas ordenadas
- Listas desordenadas
- Listas de definiciones

Primero se define el tipo de lista, y después todos los elementos. Cabe la posibilidad de "anidar" listas.

Esta es la definición de las listas ordenadas, esta ordenación viene dada por el indicativo del elemento (número o letra que antecede al elemento).

Las etiquetas <**li**> ... </**li**> definen los elementos que formarán parte de la lista. Tanto la etiqueta <**ol**>, como la <**li**> son elementos de bloque .

```
<ol>
    <li> Zona Uno </li>
    <li> Zona Dos </li>
    <li> Zona Tres </li>
</ol>
```

Los atributos que se pueden utilizar con la etiqueta <ol> son:

type – Indica el tipo de marcador que usará la lista (usar list-style-type de CSS), números, letras mayúsculas, letras minúsculas, números romanos en mayúsculas o números romanos en minúsculas.

reversed – Inicia el contador de mayor a menor (sería posible hacerlo desde CSS).

start – Indica en número por el que empezará la lista.

Utilizada para crear viñetas o listas no ordenadas. Consiste en una relación de elementos, cada uno en una fila, con un adorno al inicio.

Al igual que en las listas ordenadas, las etiquetas <**li**> ... </**li**> contienen los elementos que formarán parte de la lista. Este tipo de listas están presentes en la confección de menús.

Tanto la etiqueta <**ul**>, como la <**li**> son elementos de bloque .

```
<ul>
    <li> Elemento Uno </li>
    <li> Elemento Dos </li>
    <li> Elemento Tres </li>
</ul>
```

El siguiente es un ejemplo de anidamiento de listas, válido tanto para listas ordenadas como para listas no ordenadas

```
<ol>
    <li> Zona Uno </li>
        <li> Zona Dos
            <ol>
                <li> Zona Norte </li>
                <li> Zona Sur </li>
            </ol>
        </li>
    <li> Zona Tres </li>
</ol>
```

<dl></dl>

Las listas de definición consisten en una relación de términos con una descripción, un diccionario podría considerarse un ejemplo.

Las etiquetas <**dt**> ... </**dt**> definen los términos a describir.

Las etiquetas <**dd**> ... </**dd**> enmarcan las descripción de cada elemento. Tanto la etiqueta <**dl**>, como <**dt**> y <**dd**> son elementos de bloque.

```
<dl>
    <dt>Coffee</dt>
        <dd>Black hot drink</dd>
    <dt>Milk</dt>
        <dd>White cold drink</dd>
</dl>
```

<blockquote> </ blockquote>

Es un elemento de bloque que se emplea para citas de fuentes externas. Esta etiqueta puede llevar el atributo **cite** con la url original de la cita.

```
<blockquote cite="www.pagina.es">
        Pellentesque ipsum libero, lobortis vitae
        sollicitudin at, ultricies eu massa.
        Quisque at lobortis justo
</blockquote>
```

pre></pre>

Texto preformateado, muestra el contenido tal y como fue escrito en el documento HTML, conservando los espacios y los saltos de línea.

```
<pre>
        Pellentesque ipsum libero,
                lobortis vitae sollicitudin at,
          ultricies eu massa. Quisque at lobortis
        justo
</pre>
```


Es un elemento separador, provoca un salto de línea. Se aconseja no abusar de ella y utilizar otros métodos para provocar ese salto (mediante CSS).

Esta es una de las etiquetas que no tienen cierre. Puede verse en algunas webs con este otro formato <br \>, de la que yo personalmente no soy partidaria.

<hr>

Es otro elemento separador, pero lo hace mediante una línea horizontal. <hr \> también es otra representación.

Formato semántico de TEXTO.

En este apartado nos referiremos a elementos en línea, que no provocan salto, utilizados para dar formato o valor semantico a su contenido.

**

Texto en negrita, carece de valor semántico y se utiliza para realzar el contenido. Evidentemente esto se puede conseguir también mediante CSS.

```
<p> Contenido con <b>negrita</b> del párrafo </p>
```

**

Texto en negrita, pero con el valor semántico de reforzamiento del texto.

 Contenido importante del párrafo

```
<p> Contenido <strong>importante</strong> del párrafo </p>
```

<cite></cite>

Usado para el título de una obra, un documento o un evento, por lo tanto aporta valor semántico. Por defecto pone el texto en cursiva

<q></q>

Encierra el contenido dentro de “” (comillas) y se usa para una cita o frase extraída de otro contexto. Puede contener el atributo **cite**.

* / *

Superíndice y subíndice respectivamente

```
<p> 5<sup>2</sup> = 25 </p>
```

<mark></mark>

Usada para marcar el contenido de manera que pueda resaltarse y diferenciarse de los demás.

Si marcamos las palabras que el usuario introdujo en una casilla de búsqueda le resultará mucho más sencillo identificar el párrafo o sección donde se habla del tema buscado.

**

Esta etiqueta es comparable al div, pero como elemento de línea, no de bloque. Se utiliza para la maquetación cuando ninguna de las anteriores cumple con las necesidades.

Existen más etiqueta con significación. En el siguiente enlace hay un amplio artículo donde obtener más detalles.

https://lenguajehtml.com/p/html/semantica/etiquetas-html-de-texto

Enlaces de hipertexto.

<a>

Se trata de un elemento de línea, y define un link o hipervínculo, que se utiliza para enlazar una página a otra. La palabra o frase de enlace irá entre los tags de apertura y cierre.

También puede utilizarse como enlace una imagen. En HTML5 se permite que los contenedores de enlaces sean elementos de bloque.

Para que funcione correctamente necesita **atributos**:

href – Es un atributo imprescindible, y contendrá el destino del enlace. Este destino puede ser:

- Local a la página – dentro de la dirección debemos incluir el símbolo # y el id de la etiqueta a donde direccionamos el enlace (#inicio). Se denominan anclas internas.
- Local al sitio web – son enlaces a páginas dentro del mismo sitio web, con lo que la dirección es relativa al sitio (contacto.html)
- Externas al sitio web – aquí debemos incluir la url completa y el protocolo que se utilizará para llegar a ellas (http://google.es)
- A un email – se indica una dirección de correo, y el sistema inicia el gestor de correo electrónico que tengamos activado en nuestro equipo. El enlace se hace incorporando mailto (**'mailto:email@algo.es'**)
- A un fichero – cuando se quiere abrir un archivo (pdf, imagen, etc), se indica en la url la ubicación y el nombre del archivo a abrir.

target – permite abrir el enlace en otra ventana (**'target: _blank'**)

title – permite mostrar un texto descriptivo que se mostrará al pasar el cursor por encima (**'title: Ir a la página principal'**). Este atributo se puede usar otros elementos.

Accesskey – Especifica una tecla de acceso directo para activar o enfocar un elemento.

Dependiendo del navegador usado la combinación de teclas varía Explorer

[Alt] + accesskey

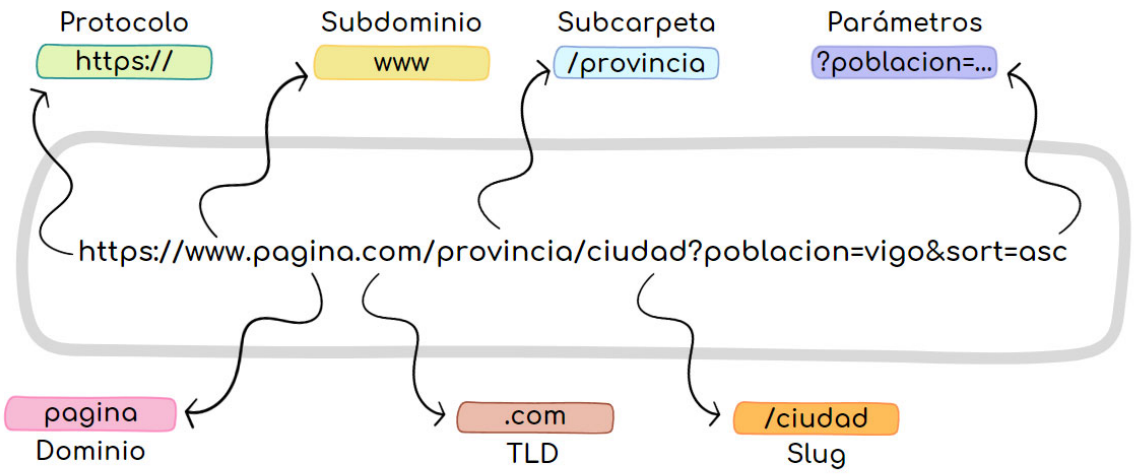
Chrome [Alt] + accesskey

Firefox [Alt] [Shift] + accesskey

Safari [Alt] + accesskey

```
<a href="http://google.es" title="Siguiente buscador"
    accesskey="g" target="_blank">
        Google, otro buscador
</a>
```

Estructura de un enlace



TLD: "Top level domain" o dominio de primer nivel, que se refiere a las extensiones:

- .**com**: webs comerciales.
- .**org**: organizaciones sin ánimo de lucro.
- .**net**: empresas de software y alojamiento.
- .**edu**: instituciones educativas (universidades, colegios, escuelas, etc.).
- .**gov**: instituciones del gobierno, como ministerios o departamentos (no en todos los países).

Slug: El slug es la parte de una URL que identifica una página específica o una entrada de una web. Ayuda a los usuarios a comprender el contexto y el contenido de una página.

Diferencias entre enlaces absolutos y relativos

Los enlaces **absolutos** son aquellos que incluyen la ruta completa (el protocolo http:// o https://, el dominio y la ruta del archivo). Apuntan a un recurso o a un lugar de internet y se usa para enlazar sitios web externos o recursos del propio sitio, pero que queremos asegurarnos que el enlace funcione sin importar desde donde se abra.

```
<a
href="https://www.ejemplo.com/curso/html/introduccion.html">Ir a
la introducción de HTML
</a>
```

Los enlaces **relativos** por el contrario no indican la ruta completa, sino que indican la ubicación del archivo en relación del archivo actual.

Se usan en sitios webs cuando las páginas están en la misma carpeta o en carpetas cercanas, por lo que son muy útiles para mantener la estructura interna del sitio sin tener que escribir la dirección completa.

```
<a href="contacto.html">Contacto</a>

    • Si el archivo esta dentro de una subcarpeta
<a href="productos/lista.html">Lista de Productos</a>

    • Si el archivo está en la carpeta anterior
<a href="../index.html">Volver al inicio</a>
```

Enlaces especiales

Enlaces para Enviar Correo (mailto)

Al hacer clic, abren automáticamente el programa de correo electrónico (por ejemplo, Outlook, Gmail en tu navegador, etc.) y preparan un nuevo mensaje con la dirección en el campo "Para".

```
<a href="mailto:correo@ejemplo.com">Enviar Correo</a>
```

- Se puede añadir el asunto

```
<a href="mailto:correo@ejemplo.com?subject=Saludo">Enviar Correo</a>
```

- E incluso la copia y el cuerpo usando ?subject=..., &cc=...,&bcc=...,&body=...

Enlaces para Llamadas de teléfono

En dispositivos con función de teléfono (smartphones), abren la aplicación para realizar la llamada directamente. Se crean usando tel: seguido del número.

```
<a href="tel:+34900111222">Llamar al +34 900 11 12 22</a>
```

- Solo funciona en móviles salvo que se use alguna aplicación VOIP

Otros enlaces - Whatsapp

Este es el tipo de enlace de mas reciente creación y que se encuentra muy de moda ahora.

Sintaxis 1

```
<a href="https://wa.me/34123456789">Envíame un mensaje</a>
```

Sintaxis 2

```
<a href="whatsapp://send?phone=34123456789">Mensaje en WhatsApp
</a>
```

Abre una ventana en whatapp para iniciar un chat con el numero que se le indica.

Además del atributo phone, podemos añadir uno de texto con un texto predefinido una opción interesante para cuando queremos diferenciar al detalle desde que parte de la web se genera el mensaje

```
Sintaxis 1
<a href="https://wa.me/34123456789?text= Aquí iría el mensaje que queramos añadir">Envíame un mensaje</a>
Sintaxis 2
<a href="whatsapp://send?phone=34123456789&text= Aquí iría el mensaje que queramos añadir ">Mensaje en  WhatsApp
</a>
```

Imágenes.

Formatos de imágenes.

GIF – Hasta un máximo de 256 colores, se reduce el tamaño de la imagen quitando colores. Permite convertir un color a transparente, se pueden hacer animaciones. No es buen formato para fotos, pero si para logos, botones, etc.

JPEG – Óptimo para fotografía digital. Hasta 16.7 millones de colores. Emplea un algorítmo de compresión que reduce el peso de la imagen, como contrapartida se produce una perdida de calidad de la imagen. Estos archivos pueden aparecer con la extensión .jpg o .jpeg.

PNG – Hasta 16.7 millones de colores. Compresión sin pérdidas, pero menos eficiente que jpeg. Permite transparencia, pero no animaciones.

WebP – Desarrollado por google, se espera que sea el formato del futuro. Permite compresión con pérdida (con un peso menor del 30% del jpeg) y sin pérdida, en este se puede usar transparencia y secuencias de imágenes (a día de hoy 04/2020) no está soportado por Safari).

SVG –Los ficheros SVG se definen en XML y permiten usar formas gráficas, mapas de bits o texto. Al mismo tiempo pueden ser estáticos o dinámicos. El escalado de las imágenes es óptimo tanto para aumentar la imagen como para disminuirla.

<img>

Permite mostrar imágenes en la web. Se trata de un elemento de línea, y no lleva tag de cierre.

Puede formar parte de un enlace o link. Atributos:

src – aquí se indica la ubicación y el nombre del archivo a visualizar. Es un atributo

obligatorio.

alt – texto alternativo que se mostrará si no existe la imagen o hay algún problema con ella. Por el tema de acesibilidad deberíamos considerarlo otro atributo obligatorio

title – texto que aparecerá al posicionar el cursos sobre la imagen.

longdesc – especifica un hipervínculo para una descripción detallada de la imagen.

```
<img src="foto.jpg" alt="Paisaje" title="Paisaje de invierno"
        longdesc="http://wiki.com#paisaje">
```

<figure></figure>

Sirve para agrupar ilustraciones, diagramas, fotos, listados de códigos, etc. Se acompaña de las etiquetas <**img**> y <**figcaption**>.

Representa un contenido de flujo que es autónomo, pudiendo alejarse del contenido primario.

```
<figure>
    <img src="foto.jpg" alt="Paisaje" title="Paisaje de invierno"
        longdesc="http://wiki.com#paisaje">
</figure>
```

<figcaption></figcaption>

Título o pie (depende del autor) de la imagen. Este bloque debe ir al inicio o al final de la etiqueta figure.

```
<figure>
    <img src="foto.jpg" alt="Paisaje" title="Paisaje de invierno"
        longdesc="http://wiki.com#paisaje">
    <figcaption>
        Pie de la foto
    </figcaption>
</figure>
```

<picture></picture>

Gestión de imágenes basada en la resolución de la pantalla utilizada. Es un contenedor para la etiqueta <**img**>, que viene apoyada en la etiqueta <**source**> que define una imagen alternativa.

La etiqueta source permite agregar uno o varios orígenes de la imagen mediante los siguientes atributos o parámetros.

media – Permite indicar el tamaño del dispositivo (deberían coincidir con los puntos de corte de las media CSS).

srcset - Nombre de la imagen .

```
<picture>
    <source media="(min-width: 1024px)" srcset="foto2.jpg">
    <source media="(min-width: 650px)" srcset="foto3.jpg">
    <img src="foto1.jpg" alt="Foto número 1">
</picture>
```

<audio></audio>

Hasta ahora el audio ha dependido de los plugins que cada usuario tenía instalados en sus equipo.

HTML5 incorpora esta nueva etiqueta para estararizar el uso de audio, aunque este estándar no está llegando a la vez en todos los navegador, y es por ello que debemos disponer de distintos formatos del mismo audio.

```
<audio controls>
    <source src="ejemplo.mp3" type="audio/mpeg" />
    <source src="ejemplo.ogg" type="audio/ogg" />
    <p>
        Su navegador no soporta elementos de audio HTML5.
    </p>
</audio>
```

WAV – Es un formato de audio digital sin compresión de datos. Desarrollado por Microsoft e IBM.

MP3 – formato de compresión de audio digital. Está sujeto a patente y licencia comercial.

OGG – fue el formato que comenzó usando Firefox, se trata de un formato libre como alternativa al MP3.

A día de hoy (04/2020) el formato MP3 está soportado por los principales navegadores.

<video></video>

HTML5 también incorpora una etiqueta para gestionar vídeo, con una estructura similar a la de audio.

Hay 3 formatos de video están soportados por la etiqueta <**video**>: MP4, WebM, y Ogg. Actualmente (04/2020) el formato MP4 es soportado por los principales navegadores.

```
<video controls>
        <source src="pelicula.mp4" type="video/mp4">
        <source src="pelilcula.ogg" type="video/ogg">
        <p>Su navegador no soporta elementos de vídeo.</p>
</video>
```

<table></table>

Una tabla <**table**> es un medio de organizar datos en filas <**tr**> y columnas <**td**>. Pueden existir también celtas de cabecera <**th**>.

El siguiente es un ejemplo en su forma más elemental :

```
<table>
   <tr>
      <th> Mes </th>
      <th> Saldo </th>
   </tr>
   <tr>
      <td> Enero </td>
      <td> 100€ </td>
   </tr>
</table>
```

Mes	Saldo
Enero	100€

Etiquetas relacionadas con las tablas:

<**tr**> ... </**tr**> – Crea una fila, puede contener uno o varios <td> y <th>

<**td**> ... </**td**>- Crea una celda de datos, es donde incluiremos el contenido

<**th**> ... </**th**>- Crea una celda de encabezado. Su ubicación normal es el la primera fila o primera columna

<**caption**> ... </**caption**>- Permite incluir un título o leyenda en la tabla. Debe ser la primera etiqueta después de <**table**>

<**thead**> ... </**thead**> - Agrupa los encabezados de la tabla (semántica)

<**tbody**> ... </**tbody**> - Agrupa el cuerpo de la tabla (semántica)

<**tfoot**> ... </**tfoot**> - Agrupa el pie de la tabla (semántica)

Los atributos de los que se disponen y que serán aplicados a los elementos <td> son:

Colspan – agrupa 2 o más celdas en la misma fila

Rowspan – agrupa 2 o más celdas en la misma columna

Formularios.

Los formularios nos permiten con el usuario final ya que nos permite rellenar información de forma ordenada y estructurada que será enviada a un correo electrónico o mejor a un servidor donde un script dinámico Web como PHP, ASP o CGI tratará la información recibida.

<form></form>

Son las etiquetas que delimitan un formulario. Puede haber varios en un mismo archivo html, pero no pueden anidarse como las listas, tablas, etc.

```
< form  id=" "  name=" "  action=" "  method=" "  enctype=" " >
</form>
```

Donde:

id y **name**: identifican al formulario, se aconseja que sean el mismo.

action - Designa la ubicación del archivo que manejará las entradas del formulario. Los formularios son usualmente manejados por scripts del lado servidor (php, jsp, asp, cgi). Si no se indica nada, el formulario se llama a si mismo.

method - Define cómo la información de un formulario es enviada al agente procesador. Existen dos valores posibles para este atributo (insensibles a mayúsculas/minúsculas):

- **get**: Los datos del formulario son agregados a la URI definida en el atributo "action" (por ejemplo, manejador.php?nombre=Juan&apellido=García), y serán visibles en

la barra de navegación.

- **post**: Los datos del formulario son agregados al cuerpo del formulario
- **Enctype** - Mediante este atributo indicaremos la forma en la que viajará la información que se mande a través del formulario, la forma puede ser de varios tipos:
- "**text/plain**" - Sólamente se convierten ls espacios en blanco en caracteres '+'.
- "**multipart/form-data**" - Este valor es obligatorio cuando utiliza formularios que tienen un control de carga de archivos
- "**application/x-www-form-urlencoded**" – Todos los caracteres se codifican antes de enviarse. Es el valor por defecto.

input

La mayoría de los controles que presentan los formularios se crean con la etiqueta **<input type="…**">, por lo que su definición formal y su lista de atributos es muy extensa.

Este atributo es imprescindible, pues indica el tipo de control de datos que se va a solicitar. Valores que puede incluir: **text**, **password**, **button**, **checkbox**, **reset**, **radio**, **hidden**, **file**, **image**, **submit**. Html5 incorpora nuevos tipos: **color**, **date**, **datetime**, **datetime-local**, **email**, **month**, **number**, **range**, **search**, **tel**, **time**, **url**, **week**.

Se aconseja utilizar en todos los elementos los atributos id y name. **Id** servirá para las validaciones en el cliente mediante Javascript, mientras que **name** será el que recibira el script en el servidor.

<input type="text" >

Se utiliza para la captura de cadenas simples de caracteres. Puede usar los siguientes modificadores:

- **NAME, ID**="texto": indica el nombre con que se referenciará a la variable asociada a dicho campo.

- **VALUE**="texto": valor por defecto.
- **SIZE**=número: tamaño del cuadro de captura. No usar, debe hacerse mediante CSS.
- **MAXLENGTH**=número: número máximo de caracteres que se capturarán.

```
<input type="text" name="nombre" id="nombre" value="valor_inicial"
maxlength="20">
```

<input type="password" >

Es muy parecido al tipo "**text**" con la diferencia de que los caracteres tecleados se sustituyen por asteriscos. Se pueden usar los mismos modificadores (aunque por supuesto no tiene sentido usar VALUE).

```
<input type="password" name="clave" id="clave">
```

<input type="radio" >

Con el tipo "**radio**" el usuario puede escoger una única opción entre varias posibilidades. Cada una de las opciones posibles usa un mismo valor para **NAME** y distinto valor para **VALUE**. El script recibirá el valor de la opción seleccionada. Puede usarse el modificador CHECKED para indicar la opción por defecto.

```
<input type="radio" name="situacion" id="alumno" value="alumno"
CHECKED> Alumno

<input type="radio" name="situacion" id="profe" value="profe">
Profesor
```

<input type="checkbox" >

Permite hacer selecciones entre diversas opciones, pero sin que sean mutuamente excluyentes. Sólo serán enviadas las opciones escogidas

- **NAME, ID**="texto": Nombre de la variable
- **VALUE**="texto": Valor que recibirá la variable cuando es escogida

- **CHECKED**: Se escoge por defecto

```
<input type="checkbox" name="cine" id="cine" value="1" CHECKED> Cine
<input type="checkbox" name="lectura" id="lectura" value="1" > Lectura
```

<input type="file" >

El tipo FILE permite que el usuario envíe un fichero al cgi. Aparece un cuadro en el que se debe escribir el nombre del fichero y un botón que permite explorar el disco del cliente. Usa los siguientes parámetros:

- **NAME, ID**="texto": nombre que identifica al campo.
- **ACCEPT**="tipoMIME": especifica el tipo de archivo.

```
<input type="file" name="fichero" id="fichero" accept="text/html">
```

<input type="submit" >

"**submit**" es un botón con un funcionamiento específico, se utiliza para enviar los datos del formulario a la **url** indicada en el parámetro **action** de la etiqueta **form**.

```
<input type="submit" name="enviar" id="enviar" value="Pulsar para enviar" >
```

<input type="reset">

El botón "**reset**" se utiliza para borrar los valores introducidos y reiniciar los campos a sus valores por defecto.

```
<input type="reset" name="borrar" id="borrar" value="Reiniciar valores" >
```

<input type="image">

El tipo "**image**" muestra una imagen en la página y cuando el usuario pincha con el ratón, se realiza el envío del formulario. Además de los datos introducidos se enviará las coordenadas del punto donde señala el ratón.

Usa los siguientes parámetros:

- **SRC**="URL": indica la ubicación de la imagen
- **NAME, ID**="texto": si no se usa, las coordenadas se identificarán como X e Y. Si se asigna un nombre, por ejemplo, NAME="coordenada", entonces se identificarán como **coordenada.X** y **coordenada.Y**.

<input type="hidden">

El tipo **hidden** define un campo de entrada oculto, que no será randerizado por el navegador. Es de especial utilidad cuando se trata de formularios de modificación, donde debemos conservar la identificación de los datos que estamos modificando.

```
<input type="hidden" name="codigo" id="codigo" value="123" >
```

<input type="button" >

HTML dispone de botones con distintas funcionalidades y utilidades. Como parámetros aceptan:

- **NAME**="texto": nombre que identifica al botón
- **VALUE**="texto": lo que aparece escrito sobre el botón.

```
<input type="button" name="boton" id="boton" value="Pulsar">
```

Como se intuye, el tipo "**button**" presenta un botón, pero debe controlarse usando javascript para dotarlo de funcionalidad.

<button></button>

La etiqueta de HTML <**button**> representa un elemento cliqueable de tipo botón que puede ser utilizado en formularios o en cualquier parte de la página que necesite un botón.

```
<button name="boton" id="boton" >Pulsar</button>
```

Atributos:

- **TYPE** = Indica el tipo de botón o función, puede tomar los siguientes valores button, reset o submit (valor por defecto).
- **FORMTARGET** = solo con el type="**submit**" y funciona como el target de los links.

<input type="email">

El tipo **email** se emplea para correos electrónicos. La principal novedad es que realiza una verificación de que el formato de la cuenta de correo electrónico es correcto (sólo la hace si se introduce algún valor), aunque requiere de una verificación o validación mas concreta.

```
<input type="email" name="correo" id="correo">
```

<input type="url">

El tipo **url** realiza la verificación de que el formato de la url es correcto (sólo la hace si se introduce algún valor).

```
<input type="url" name="web" id="web">
```

<input type="color">

Permite crear un campo en el cual se desplegará una paleta para que el usuario pueda elegir un color. Aún no está implementado por todos los navegadores

<input type="number">

Permite crear un campo de entrada para números. Aún no está implementado por todos los navegadores.

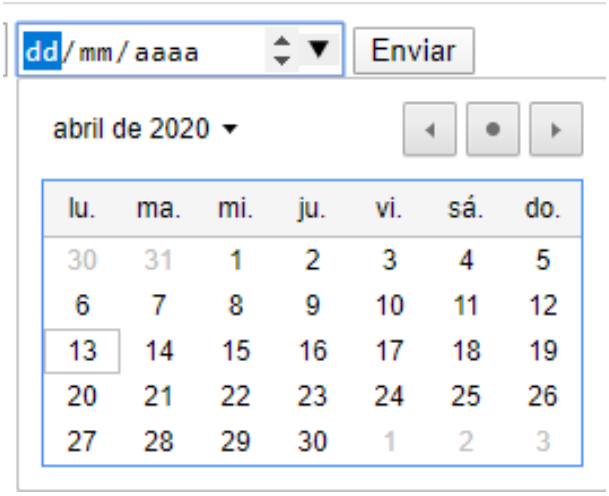
- **MIN** y **MAX** = Fijan los valores máximos y mínimos que puede tener el input.
- **STEP** = Valor numérico con el cual varía el intervalo del input (step="2", sumarías o restarías de 2 en 2 dando a las flechas), se puede utilizar decimales.
- **READONLY** = Con el valor activado el input no podrá variar de valor, no se podrá editar (readonly="readonly").

```
<input type="number" name="edad" id="edad" min="18" max="99" step="5" >
```

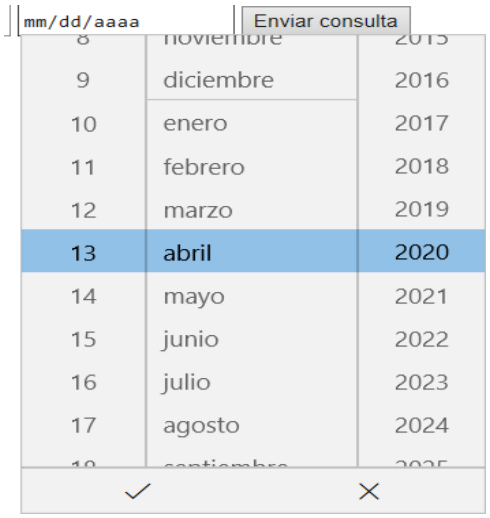
<input type="date">

Con este valor, es posible crear un campo que desplegará un calendario en el cual se podrá seleccionar una fecha, de la cual se podrá obtener el año, el mes y el día.

Así se ve en Firefox cuando se posiciona el curso sobre el elemento



Esta sería la representación en Chrome y Opera. En este navegador hay que pulsar sobre la flecha para ver el desplegable



Por lo que respecta a Safari de Apple, no hace una representación visual como sus competidores, al menos en la versión para Windows.

Muestra las flechas con las que se puede subir o bajar de fecha.

Representacion de Edge de Microsoft

<input type="datetime-local">

Similar al anterior, pero para la hora local. Aún no está implementado por todos los navegadores.

<input type="month">

Este elemento nos brinda la posibilidad de desplegar un calendario para elegir año y mes. Aún no está implementado por todos los navegadores.

<input type="week">

Permite desplegar un calendario para elegir una semana (el número de semana de un determinado año). Aún no está implementado por todos los navegadores.

<input type="time">

Debemos tener en cuenta que este elemento se puede utilizar para seleccionar minuto y segundo. Aún no está implementado por todos los navegadores.

<input type="range">

Este tipo nos permite crear un control de entrada que mostrará un deslizador (slider) para seleccionar un valor numérico, al igual que **number**, puede trabajar con los atributos **min**, **max** y **step**, Aún no está implementado por todos los navegadores.

<input type="search">

Es un tipo de campo que puede resultarnos de utilidad, por su contenido semántica, para incluir en nuestro proyecto una caja de búsqueda.

<input type="tel">

Campo destinado para números de teléfono.

<textarea></textarea>

Con "**textarea**" definimos un texto de múltiples líneas para captar textos grandes. Si entre la apertura y cierre de esta directiva se escribe algo, ese texto aparece escrito por defecto.

- **NAME, ID** ="texto": nombre de la variable
- **ROWS** =número: número de filas del cuadro. Usar CSS
- **COLS** =número: número de columnas. Usar CSS
- **MAXLENGTH** = puede recibir como valor un número que indique el máximo de caracteres para esta área de texto.

```
<textarea name="comentario" id="omentario" placeholder="Deje su comentario"></textarea>
```

<select></select>

Permite realizar selecciones en forma de menú desplegable. La sintaxis difiere bastante de la de los tipos anteriormente descritos. En este caso, se usa una directiva llamada

"**select**" entre cuya apertura y cierre se escriben la etiqueta "**option**" con cada una de las posibles opciones que se deben mostrar. Dentro de select podemos incluir:

- **NAME, ID** = "texto": nombre de la variable
- **MULTIPLE** = permite selección de varias opciones
- **SIZE** = número: fija el número de líneas del cuadro de selección que serán mostradas.

Por su parte, en option indicaremos los valores seleccionables, con el título que se mostrará entre las etiquetas de apertura y cierre y el valor que se enviará al servidor.

```
<select name="opcion" id="opcion" >
        <option value="1" selected> Básica </option>
        <option value="2"> Parcial </option>
        <option value="3"> Total </option>
</select>
```

<datalist>

Un **dataList** en HTML5 es una lista de opciones predefinidas que se le pueden pasar al elemento INPUT. Podemos aplicarlo al tipo **text**, al tipo **url** o al tipo **range**. Para asociar el input con la lista se utilizan los atributos **list** e **id**.

```
<input list="listacolores" name="colores" id="colores">
 <datalist id="listacolores">
        <option value="Azul">
        <option value="Rojo">
        <option value="Amarillo">
        <option value="Negro">
        <option value="Blanco">
</datalist>
```

Otros atributos

Además de los atributos vistos en las anteriores etiquetas, que se pueden considerar específicos, podemos destacar los atributos:

- autocomplete: permite indicar si el campo podrá tener la característica de autocompletado (valor on) o no (valor off).
- autofocus: el campo que reciba este atributo tendrá el foco al ser cargado el formulario con el cual estamos trabajando.
- placeholder: nos permite especificar un texto que será el que se muestra en el campo del formulario cuando no está enfocado y el usuario todavía no escribió sobre dicho control.
- pattern: se trata de un interesante atributo que nos permite especificar un patrón para el campo correspondiente.
- required: se utiliza para indicar que el campo debe ser completado antes de que el formulario sea enviado

Cascading Style Sheets, CSS es un lenguaje usado para definir la presentación de un documento escrito en HTML.

El uso de HTML junto con CSS es la manera más adecuada de diseñar un sitio Web. Permite separar el contenido y la presentación, con lo que además se consigue facilitar el mantenimiento del aspecto visual del sitio web.

Con CSS podemos controlar numerosas propiedades del texto, entre ellas las fuentes, negrita, cursiva, subrayado y sombras de texto; color de texto y de fondo; color y subrayado de vínculos; etc. Si utilizamos CSS para controlar las fuentes, también conseguirá un tratamiento más coherente del diseño y el aspecto de la página en múltiples navegadores.

También se utiliza para definir el formato y la posición de elementos que componen nuestro documento.

Hay que tener en cuenta qué si existen 2 definiciones de una misma propiedad sobre un elemento, la última es la que prevalece.

Tipos de Hojas de Estilo

Hay 3 formas de insertar una hoja de estilos en un documento:

- CSS integrado en el elemento.
- CSS interno al documento.
- CSS externo al documento.

CSS integrado en el elemento

También llamado estilo entre líneas.

Se aplica directamente sobre las etiquetas haciendo uso del atributo HTML **style**, indicándole para cada estilo la propiedad: valor. Si queremos aplicar varios estilos a una etiqueta, usaremos una sola vez el atributo style, separando cada par **propiedad : valor** con un punto y coma ";"

```
<p style="color: red; font-size: 1.5em;">
        Texto del párrafo
</p>
```

Las definiciones a nivel de línea son las últimas en aplicarse, por lo que son las que prevalecen en caso de duplicidades

Esta forma de utilización está **TOTALMENTE** desaconsejada, pues cualquier corrección entraña una modificación línea a línea.

Hojas de estilo internas

Se definen en el apartado **head** del código HTML, y va integrado entre las etiquetas <**style**> y </**style**>.

En este caso debemos indicar a qué elemento (**selector**) queremos aplicarle el estilo y para cada estilo tenemos que utilizar el par **propiedad : valor** (acabado en punto y coma).

```
<style>
    p {
        color: red;
        font-size: 20px;
    }
</style>
```

De esta forma aplicamos el mismo estilo al elemento indicado (en este caso todos los párrafos serían rojos de 20 píxeles).

Este modelo de inserción de estilos e adecuado cuando sólo se va a utilizar en una página y la cantidad de estilos es pequeña.

Hojas de estilo externas

Se trata de un fichero independiente con las definiciones CSS necesarias. Debemos agregarlo a nuestra web en el head mediante la instrucción:

```
<link href="nombreArchivo.css" type="text/css" rel="stylesheet" >
```

Dentro de este fichero, con extensión css, hay que indicar el selector al que aplicarle el estilo y para cada estilo tenemos que utilizar el par propiedad: valor (acabado en punto y coma).

```
p {
    color: red;
    font-size: 20px;
}
```

La utilización de hojas de estilo externas es la forma ideal cuando queremos definir un sitio entero con la misma estructura, pues podemos usar el mismo fichero CSS para todas nuestras páginas.

Es la forma óptima de incluir los estilos CSS. Pueden convivir 2 o más hojas de estilo dentro de un documento HTML, aunque para optimizar la carga de la web se aconseja no abusar de la cantidad de archivos empleados.

Reglas y selectores

Las reglas se componen de 2 partes, la **propiedad** que queremos cambiar y el **valor**, separados por el símbolo de dos puntos "**:**" y finalizado por punto y coma "**;**".

Selectores son los elementos sobre los que se aplicará el estilo, y que podrá contener varias reglas. Pueden ser:

- El nombre de las **etiquetas** HTML
    - Etiquetas de elementos HTML (p, header, span, etc).

```
section { }
```

- Nombres de **clases**
    - Se emplean para generar reglas que se van a usar en varios sitios.
    - El desarrollador decide el nombre que tendrá.
    - Este nombre solo contendrá letras, números, guion bajo. No podrá empezar por número.
    - Distingue entre mayúsculas y minúsculas (Pepe ¡= pepe)
    - Para definirlo se marca con el punto "." .

```
.verde { }
```

- **Identificadores**
    - Se emplean para generar reglas que se van a usar en un solo sitio de la página.
    - El desarrollador decide el nombre que tendrá.
    - Este nombre solo contendrá letras, números, guion bajo. No podrá empezar por número.
    - Distingue entre mayúsculas y minúsculas (Pepe ¡= pepe)
    - Para definirlo se marca con el símbolo # .

```
#caja1 { }
```

Una vez decidido que elemento vamos a modificar debemos agregar las reglas entre llaves **{ }**. Es muy importante tener en cuenta que no se pueden mezclar o cruzar las llaves de 2 elementos.

Existen selectores avanzados que aparecieron con CSS 2.1 y nos permiten afinar más a la hora

de aplicar css.

Selector descendente

Se utiliza para indicar un elemento que está dentro de otro, sin importar la profundidad a la que se encuentre.

```
div  article { }
```

```
<div>
    <article>
    </article>
</div>
```

```
<div>
    <section>
        <article>
        </article>
    </section>
</div>
```

Selector de hijo

Similar al selector descendente, pero en este caso el elemento dependerá directamente del selector externo. Se utiliza el signo de mayor que (**>**) entre ambos selectores.

```
div  >  article { }
```

```
<div>
    <article>
    </article>
</div>
```

Selector adyacente

Se trata de 2 elementos hermanos que están pegados el uno al otro. Emplearemos el símbolo de sumar (**+**).

h1 + h2 { }	
* Título 1 * * Título 2 *	* Título 1 * * Título 2 * * Título 3 * *Título 3 no se vería afectado*

Selector hermano

Se trata de 2 elementos hermanos que NO necesitan estar pegados el uno al otro, pero si al mismo nivel. Emplearemos el símbolo (~).

```
h1 ~ h2 { }
```

```
<h1> Título 1 </h1>
<h2> Título 2 </h2>
```

```
<h1> Título 1 </h1>
<h2> Título 2 </h2>
<h2> Título 3 </h2>
```

Título 3 también se vería afectado

Selector de atributos

Permite seleccionar elementos html en función de sus atributos y/o del valor de estos. Supongamos el caso de un **div.**

- **div[nombre_atributo]** -> Afectaría a los divs que tengan un determinado atributo.

```
div[class=""]{ }
```

```
<div>
    <div class="verde">
    </div>
</div>
```

```
<div>
    <div class="rojo">
    </div>
</div>
```

- **div[nombre_atributo = ”valor”]** -> Afectaría a los divs que tengan un determinado atributo con un valor concreto.

```
div[class="verde"]{ }

    <div>
        <div class="verde">
        </div>
    </div>
```

- **div[nombre_atributo ~= ”valor”]** -> Un atributo puede tener varios valores.

```
div[class ~= "verde"]{ }
 <div class="roja">
        Caja 1
 </div>
<div class="roja verde azul">
        Caja 2
</div>
```

*Afectará a todos los divs que entre los valores del atributo class figure **verde**. En este caso se vería afectada Caja 2.*

- **div[nombre_atributo |= ”valor”]** -> Afectaría a los divs que tengan en un determinado atributo una lista de palabras separadas por un guion y que empiece por un valor indicado.

```
div[class |= "verde"]{ }
    <div class="verde-roja">
        Caja 1
    </div>
    <div class=" verde azul">
        Caja 2
</div>
```

*Afectará a Caja 1, que es el que cumple el patrón "**verde-**" como inicio del nombre de la clase.*

Este es un patrón muy utilizado con los iconos generados a través de fuentes.

- **div[nombre_atributo ^= "valor"]** -> Afectaría a los divs que tengan un determinado atributo que empiece por un valor indicado.

```
div[class ^= "verde"]{ }
    <div class="verde_roja">
        Caja 1
    </div>
    <div class=" verdeazul">
        Caja 2
    </div>
```

*Afectará a los 2 divs, ya que cumplen el patrón "**verde**" como inicio del nombre de la clase.*

- **div[nombre_atributo $= "valor"]** -> Afectaría a los divs que tengan un determinado atributo que acabe por un valor indicado.

```
div[class $= "claro"]{ }
    <div class="verde_claro">
        Caja 1
    </div>
    <div class=" rojo_claro">
        Caja 2
    </div>
```

*Afectará a los 2 divs, ya que cumplen el patrón "**claro**" como final del nombre de la clase.*

- **div[nombre_atributo *= "valor"]** -> Afectaría a los divs que tengan un determinado atributo que contenga una ocurrencia de un valor indicado.

```
div[class *= "decla"]{ }
    <div class="verdeclaro">
        Caja 1
    </div>
    <div class=" declarar">
        Caja 2
    </div>
```

*Afectará a los 2 divs, ya que aparece el patrón "**decla**" dentro del nombre de la clase.*

Pseudo clases

Permiten modificar el comportamiento de una etiqueta en función de algunos estados especiales o usos de ellas.

:link

Se usa para dar formato a enlaces que no hayan sido visitados.

:visited

Se usa para dar formato a enlaces que ya han sido visitados.

:hover

Se usa para dar formato a enlaces cuando el ratón pasa por encima.

:active

Se usa para dar formato a enlaces cuando se presiona sobre él.

:focus

Se usa para dar formato cuando un elemento recibe el foco.

Pseudo elementos

Son elementos que se añaden a los selectores de CSS, y que a diferencia de las **pseudo- clases** que nos permitían seleccionar estados, ahora permite seleccionar una parte concreta del documento.

::first-line

Permite seleccionar la primera línea de texto de un elemento.

::first-letter

Permite seleccionar la primera letra de texto de un elemento.

::before y ::after

Se utilizan en combinación con la propiedad **content** de CSS para añadir contenidos antes o después del contenido original de un elemento

::selection

Se produce cuando el usuario hace una selección de texto con el ratón, permitiendo que se cambien las propiedades a ese trozo de texto seleccionado.

:nth-child(numero)

Selecciona el enésimo elemento hijo. Pueden utilizarse como índices los valores **odd** y **even** para indicar elementos impares (odd) o pares (even). También se puede utilizar fórmulas del

estilo (3n+2). El primer elemento es el 0 y se considera par.

:nth-last-child(numero)

Selecciona el enésimo elemento hijo contando desde el último elemento.

:first-child, :last-child

Selecciona el primer o último elemento hijo respectivamente.

Cascada y herencia

El concepto de **cascada** se refiere al orden en que se importan las reglas CSS: cuando dos reglas tienen la misma especificidad, se aplica la que viene en último lugar en el CSS.

1. Se toman las propiedades por defecto del navegador.
2. Se toman las hojas de estilo externas.
3. Hojas de estilo internas.
4. Estilos en línea.

La **especificidad** es el modo que tiene el navegador de decidir qué regla se aplica si diversas reglas tienen selectores diferentes pero podrían aplicarse a un mismo elemento. Básicamente, la especificidad mide cuán específica es la selección de un selector:

Un selector de elementos es **menos específico** si selecciona todos los elementos de aquel tipo que aparecen en la página; entonces presenta una puntuación más baja en especificidad.

Un selector de clase es **más específico** si selecciona solo los elementos de una página que tienen un valor de atributo class específico; entonces recibe una puntuación mayor.

La **herencia** se refiere a los valores de las propiedades CSS que se han establecido para los elementos padre y que los heredan los elementos hijo, pero otros no.

La siguiente es una relación de propiedades que se pueden heredar. Para que no sean heredadas se le puede aplicar la palabra clave **initial**.

color, cursor, empty-cells, font-family, font-size, font-style, font-variant, font-weight, font, letter-spacing, line-height, list-style-image, list-style-position, list-style-type, list- style, text-align, text-indent, text-transform, visibility, white-space, word-spacing.

Unidades de medida y colores

Las unidades de medida pueden ser:

- **Absolutas**: que no se adaptan a los dispositivos. Su tamaño no cambia y es siempre fijo. No siguen las tendencias actuales de diseño y tampoco son adecuadas para la

accesibilidad. (gente que necesita ver la pantalla en dimensiones más grandes por problemas de visión o similares). Píxel, centímetro, punto, pulgada.

- **Relativas**: son unidades que dependen de algún otro factor. Son adaptables a diferentes dispositivos. Dentro de estas unidades relativas cabe destacar las unidades **viewport**, que son las relativas a la ventana de visualización.
  - **rem**: Estas unidades son relativas a la fuente raíz de la web. 1rem = al tamaño de la fuente raíz (normalmente 17px). Es una buena unidad para webs adaptables porque permite relacionarlo todo a una misma cantidad y podemos escalar fácilmente toda la web. Si se desea modificar la fuente base, se puede hacer sobre la etiqueta css "**html**" para que lo herede todo el documento. No depende de la definición del elemento padre. Admite decimales.
  - **em**: Es parecida a la anterior, pero es relativa al tamaño fuente del elemento en que esté (que si no está definido se hereda del superior, del padre). Es un tamaño que se ha usado mucho, pero puede llevar a confusiones porque las fuentes dependen de unos elementos que pueden depender de otros. Admite decimales.
  - **%**: Porcentaje. Relativo al elemento padre.

Dentro de las unidades relativas existen unas especiales que son relativas a la ventana de visualización (Unidades relativas **viewport**) y por lo tanto pensadas para su uso en diferentes dispositivos:

- **vw**: viewport width. – 1vw = 1% del ancho del navegador.
- **vh**: viewport height. – 1vh = 1% del alto del navegador. Solo funciona en dispositivos móviles, en desktop tiene unos comportamientos un poco erráticos.

Estas unidades permiten poner como referencia el tamaño de la pantalla, algo importante en un mundo de dispositivos de tamaños cambiantes.

Respecto a los **colores** en CSS se pueden utilizar distintos formatos:

- con los nombres en inglés (red, green, etc.).
- con la notación hexadecimal (#ff0000, #ffff00). Existe también la notación hexadecimal abreviada (#f00, #ff0), este sistema usa cifras duplicadas, por lo tanto, no permite la misma cantidad de combinaciones.
- RGB, cantidad de rojo, verde y azul rgb(255,0,0).
- HSL, matiz, saturación y luminosidad hsl(0, 100%, 50%).
- tanto el RGB como el HSL incorporan un nuevo valor indicativo de opacidad, que en ambos casos va desde 0 a 1.

Propiedades tipográficas

CSS define numerosas propiedades para modificar la apariencia del texto y permite aplicar estilos complejos y muy variados al texto de las páginas web.

color

Indica cual será el color de la letra aplicando las notaciones vistas anteriormente.

font-family

Familia a la que pertenece la fuente (arial, helvetia, …).

Se pueden indicar varias familias separadas por comas. Si la primera fuente no se puede cargar, se utiliza la segunda, y así sucesivamente.

También se puede usar el nombre genérico (serif, sans-serif, cursive, fantasy o monospace).

En principio el navegador utiliza las fuentes que existen en el ordenador del usuario, con lo qué, si no existe, no se podrá utilizar. Existen repositorios en internet que proporcionan gran cantidad de fuentes, el más reconocido es Google Fonts.

Css dispone de la regla **@font-face** que permite agregar fuentes, haciendo una llamada a los ficheros de definición de fuentes.

```
@font-face {
        font-family: "nombre_fuente",
                        url ("nombre_fichero_fuente.extension");
}
```

font-size

Establece el tamaño de la fuente. Puede usarse medidas relativas o absolutas.

font-weight

Grosor o peso del trazo, vulgarmente conocida por **negrita**. Este grosor puede definirse numéricamente (100, 400, …, esto depende de las características de la fuente utilizada), aunque normalmente se utilizan los valores normal y bold.

font-style

Inclinación del texto

- oblique: la forma de la letra no cambia, simplemente se inclina un poco. No todas las fuentes la permiten.
- italic: en castellano se denomina cursiva, y es una versión especial de la fuente con una inclinación.
- Normal: desactiva las otras dos.

font-variant

Admite los valores normal y **small-caps**, éste último convierte el texto a mayúsculas, pero

con el tamaño de las minúsculas (Versalitas).

line-height

Permite establecer el interlineado de un bloque de texto. Los valores 100% o 1em indican interlineado sencillo.

text-align

Marca el alineamiento DENTRO de un elemento de bloque o celda de una tabla. Permite los siguientes valores.

- Left
- right
- center
- justify

Aunque por defecto el texto está alineado a la izquierda, esta es una de las propiedades que se heredan de los contenedores padre.

text-decoration

Añade o quita decoración a un elemento. Puede llevar varias:

- underline – Subraya el texto.
- overline – El subrayado aparece en la parte superior del texto.
- line-through – El texto se muestra tachado.
- none – Quita todas las decoraciones.

Esta propiedad se suele manejar a la hora de gestionar los enlaces o links.

text-indent

Establece la tabulación o sangría de la primera línea de un elemento.

word-spacing

Establece la separación entre las palabras del texto. Admite valores negativos. El valor clave **normal** es el que tiene por defecto, e indica que el navegador decide dicha separación.

letter-spacing

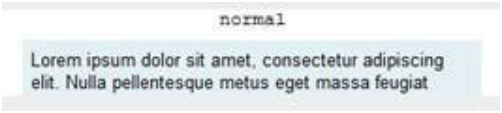
Establece la separación entre letras. Admite valores positivos, negativos y el valor clave

normal.

white-space

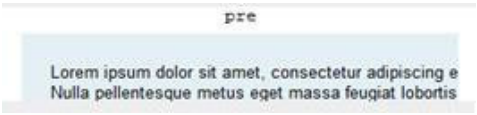
Determina cómo se maneja el espacio en blanco dentro de un elemento con una de las siguientes palabras:

- normal – Las secuencias de espacios en blanco son reducidas a un solo espacio.



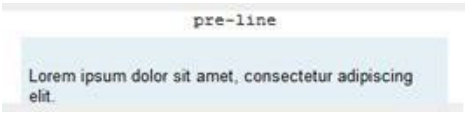
- nowrap – Reduce espacios en blanco igual que el modo normal, pero suprime saltos de línea del origen.

- pre – Secuencias de espacios son preservados. Líneas son solo rotas en caracteres de saltos de línea encontrado en el origen y en elementos html
.



- pre-wrap – Secuencias de espacio son preservadas. Líneas son rotas en caracteres de saltos de línea, en elementos html
, y agrega saltos necesarios para rellenar los cuadros de línea.

- pre-line – Secuencias de espacios en blanco son reducidas. Líneas son rotas en caracteres de salto de línea, en elementos html
, y los necesarios para rellenar los cuadros de línea.



Fondos

background-color

Indica cual será el color de fondo del elemento.

background-image

Establece una imagen como fondo del elemento. Si la imagen tiene transparencias, se verá el color de fondo.

```
background-image: url(carpeta/imagen.ext);
```

background-repeat

Indica si la imagen se repetirá para cubrir todo el fondo. Los valores posibles son:

- **repeat**: se fuerza la repetición de la imagen para rellenar el contenedor.
- **Repeat-x**: sólo se repite para llenar la primera fila.
- **repeat-y**: Se repite en la primera columna.
- **no-repeat**: sólo aparece una vez la imagen.

background-size

Permite establecer el tamaño de la imagen de fondo. Los valores que admite son:

- **auto**: el primer valor corresponde al tamaño horizontal y el segundo al tamaño vertical, se conserva el tamaño original de la imagen.
- **valores numéricos** (porcentajes o distancias): el primer valor corresponde al tamaño horizontal y el segundo al tamaño vertical, la imagen se deforma para ajustarse a esos valores.
- **contain**: la imagen se aumenta o disminuye de manera que pueda verse completa en una de las dimensiones (horizontal o vertical) y se repite en la otra dimensión.
- **cover**: la imagen se aumenta o disminuye de manera que ocupe todo el espacio disponible en una dimensión (horizontal o vertical) y se corta en la otra dimensión.

background-attachment

Establece el comportamiento de la imagen de fondo cuando se desplaza el elemento (scroll de pantalla).

- **scroll:** la imagen se desplaza con el contenedor.
- **fixed:** la imagen permanece fija.

background-position

Posición horizontal y vertical de origen de la posición imagen fondo. Los valores que admite son:

- **valores numéricos** (porcentajes o distancias).
- **left**, **center** y **right**: establecen la posición horizontal (izquierda, centro y derecha, respectivamente).
- **top**, **center**, **bottom**: establece la posición vertical (arriba, en medio, abajo, respectivamente).

```
background-position: top left;
```

background-origin

Posición de inicio del fondo.

- **border-box**: el origen está en el borde (borde incluido).

- **padding-box**: el origen está en el margen interior (el borde no se incluye).
- **content-box**: el origen está en el contenido (no se incluye ni el margen interior ni el borde).

background-clip

Zona en la que aparece el fondo (color o imagen).

- **border-box**: el origen está en el borde (borde incluido).
- **padding-box**: el origen está en el margen interior (el borde no se incluye).
- **content-box**: el origen está en el contenido (no se incluye ni el margen interior ni el borde).

Gradientes

Un gradiente es una transición entre colores (degradado), en css trabajaremos con dos tipos:

linear-gradient – Es una transición lineal. Una vez aplicado el color de fondo, se deben indicar los parámetros dirección, color-stop1, color-stop2, ..., siendo color-stopX los colores que se utilizarán.

```
background: linear-gradient(red, yellow); /* Standard syntax */
```

Hay que tener en cuenta la existencia de navegadores antiguos, por lo que debemos añadir la sintaxis vieja utilizando los prefijos para cada tipo de navegador.

Degradado de arriba hacia abajo (*es el valor por defecto*).

```
/* Para Safari */
        background: -webkit-linear-gradient(red, yellow);
/* Para Opera */
        background: -o-linear-gradient(red, yellow);
/* Para Firefox */
        background: -moz-linear-gradient(red, yellow);
```

De izquierda a derecha.

```
background: -webkit-linear-gradient(left, red , yellow);
background: linear-gradient(to right, red , yellow);
```

Diagonal : Debe indicarse la posición de inicio horizontal y vertical.

```
background: -moz-linear-gradient(bottom right, red, yellow);
background: linear-gradient(to bottom right, red, yellow);
```

Angulares: linear-gradient(*ángulo, color-stop1, color-stop2*); El ángulo puede ser positivo o negativo, y debe indicarse la unidad de medida (deg).

```
background: -o-linear-gradient(-90deg, red, yellow);
background: linear-gradient(-90deg, red, yellow);
```

Con transparencias: Debe utilizarse el formato de color que incluye la opacidad (rgba o hsla), y funcionan como los vistos hasta el momento.

radial-gradient. Se refiere a gradientes con formas circulares o elípticas**.**

```
background: radial-gradient(parámetros)
```

Los parámetros utilizables son:

o Posición inicial – Punto central, que viene dada por una o dos coordenadas. Si se omite se considera que el inicio se corresponde con el punto central.

o Forma y dimensión o tamaño – Si utilizamos forma y tamaño, este último no es obligatorio, debemos optar por las claves circular o elipse para definir la forma, y para la dimensión disponemos de las claves losest-side indica que el círculo o elipse debe crecer hasta el lado más cercano. La palabra farthest-corner indicaría que debe crecer hasta la esquina más lejana. Contain sería lo mismo que decir closest-side y cover sinónimo de farthest-corner.

o Si no utilizamos el binomio forma/dimensión, podemos indicar el tamaño que adquirirá mediante un par de medidas (si las 2 son iguales se trata de una forma circular).

o Colores – Indicaremos todos los colores que se desean utilizar separándolos por comas.

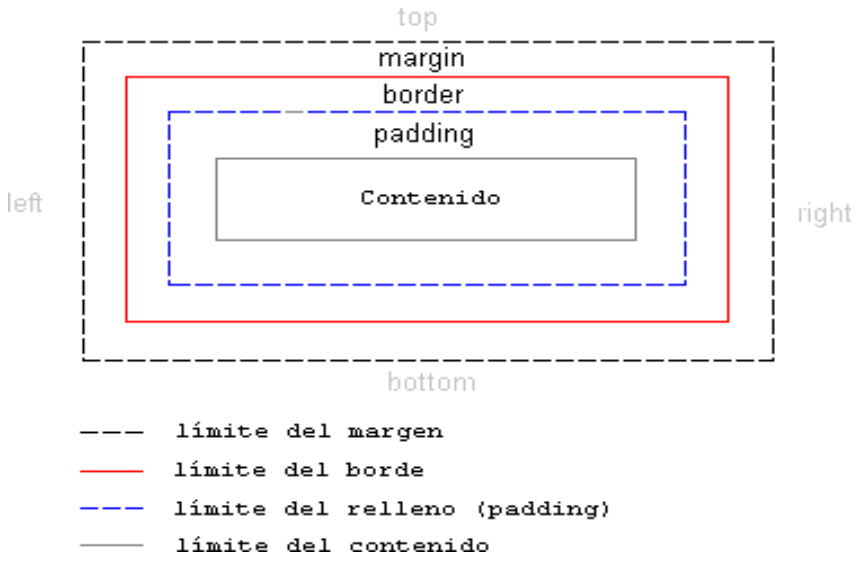
```
background: radial-gradient(ellipse cover, #66f, #f80, #ffc);
background: -webkit-radial-gradient(ellipse cover, #66f, #f80, #ffc);
background: -moz-radial-gradient(ellipse cover, #66f, #f80, #ffc);
```

Modelo de Caja

Cada elemento que encontramos dentro de un documento HTML se encuentra contenido en una caja rectangular que cuenta con una serie de propiedades que afectarán el cómo se muestran los elementos y su relación con los elementos adyacentes.

Partiendo de la parte interior de la caja se encuentran los siguientes elementos:

- **Contenido** – lo que se quiere mostrar, normalmente textos u otras cajas
- Margen interior o **padding** – es la distancia desde el contenido al borde
- **Borde** – parte exterior de la caja
- **Margen** exterior – distancia entre la caja y los elementos externos adyacentes



width

Indica el ancho del elemento, puede darse en medidas absolutas -px- o relativas -%- (min-width, max-width).

height

Determina el alto. Igualmente se utilizan medidas absolutas o relativas (min-height, max-height).

padding

Es la distancia interior entre el borde y el contenido. Se puede gestionar cada lado independientemente (padding-top, padding-right, etc) o de forma agrupada. Los tamaños pueden ser en medidas absolutas o relativas positivas:

- Si solo se da un valor- se refiere a los 4 lados.
- Si se dan 2 valores – el primero indica el superior y el inferior, y el segundo se refiere al derecho e izquierdo.
- Si se dan 3 valores – el primero indica el superior, el segundo se refiere al derecho e izquierdo y el tercero al inferior.
- Si se dan 4 valores – funciona como las agujas del reloj partiendo de la parte superior (top, right, bottom y left).

margin

Se refiere a la distancia exterior entre la caja los elementos anexos a ella. Funciona como el padding a la hora de recibir valores, con la salvedad de que ésta admite valores negativos.

border

Es el borde de la caja y puede indicarse cada lado independientemente.

- border-size – Tamaño del borde
- border-color- Color
- border-style – formato del borde. Los valores más usuales son:
    - none: No se muestra ningún borde.
    - solid: Borde continuo, formado por una línea recta continua.
    - dotted: Borde punteado.
    - dashed: Borde discontinuo.
    - double: Borde doble, formado por dos líneas rectas continuas separadas entre sí por un espacio en blanco (ancho mínimo 3px).

Estas propiedades pueden aplicarse individualmente o las tres conjuntamente (el orden no influye)

border-radius

Permite generar bordes redondeados dando el tamaño del ángulo en unidades relativas o absolutas. Puede darse un solo valor, lo que afectaría a los cuatro extremos o

individualmente.

box-sizing

Varia como se contabilizan el borde y el padding de una caja

- **border-box** – width y height indican el tamaño del contenido, más el borde y el padding, (no incluye los margin)
- **content-box** – Es el valor por defecto. Las propiedades width y height incluyen sólo el contenido. Borde, relleno o margen no están incluidos.

box-shadow

Agrega sombra a una caja. Se indicará:

- color de la sombra
  - En caso de no declarar un valor del color, toma el de la propiedad color del elemento.
- Desplazamiento horizontal de la sombra (offset)
  - Un valor positivo: sombra a la derecha.
  - Negativo: hacia la izquierda
- Desplazamiento vertical:
  - Aplica lo mismo que en el anterior, pero en la vertical. valor positivo hacia abajo, negativo hacia arriba.
- radio de desenfoque
  - "Blur Radius". Es opcional. El radio de desenfoque define la dureza o suavidad de la sombra. Cuanto menor sea el valor, más nítida la sombra. Un valor alto creará un desenfoque grande. Admite cualquier valor, pero los negativos son computados como Cero (0).
- Distancia de propagación (opcional).
  - Otro valor de la longitud, puede ser positivo o negativo, y define el tamaño de la sombra respecto al del elemento. Un valor negativo hará que la sombra sea más pequeña que el elemento que la genera y un valor positivo mayor.
- Inset/outset (opcional) Marca hacia dónde se creará la sombra
  - Inset: se crea en el interior del elemento
  - outset por fuera de él.
  - Se puede declarar al inicio o al final, si se intercala entre los demás valores se anula la declaración.

Overflow, overflow-x, overflow-y

Controla el comportamiento de los contenidos que no caben en su elemento contenedor.

visible – Es el valor por defecto, hace que los contenidos se salgan del elemento y sean completamente visibles.

hidden – los contenidos que no quepan en la caja no se muestran. Este comportamiento se produce cuando el contenido no es otro bloque, si el contenido es un bloque, la caja contenedora crece con el contenido.

scroll – Si el contendor tiene tamaño fijo, el contenido será recortado, pero se agrega una barra de desplazamiento para ver el resto del contenido. Si no tiene un tamaño fijo, el contenedor crece y muestra las barras de scroll.

auto – Se agrega una barra de desplazamiento para ver el resto del contenido cuando sobrepasa el tamaño del contenedor.

Overflow-x, overflow-y funcionan como overflow, pero sólo sobre el eje X y el eje Y respectivamente.

Visibility

Permite hacer visibles o invisibles las cajas de los elementos.

visible – Es el valor por defecto, hace visible el elemento.

hidden – Oculta el elemento, pero mantiene el espacio que ocupa.

collapse – sólo se utiliza en tablas con filas y columnas ocultándolas. Fuera de las tablas funciona como hidden.

Float

Flota elementos permitiendo que los siguientes fluyan alrededor de él. La propiedad **clear** corta la posibilidad de flotados.

left – Desplaza el contenedor a la primera posición libre a la izquierda.

right – Desplaza el contenedor a la primera posición libre a la derecha.

none – Es el valor por defecto.

Clear

Se utiliza para romper la secuencia de flotados, esto es, cuando queremos que a partir de un elemento no se flote más.

left – El elemento si va flotado a la izquierda genera una nueva línea, no busca hueco con los anteriores elementos flotados a la izquierda. Si el elemento va flotado a la derecha, si buscará al anterior elemento flotado a la derecha.

right – El elemento si va flotado a la derecha genera una nueva línea, no busca hueco con los anteriores elementos flotados a la derecha. Si el elemento va flotado a la izquierda, si buscará al anterior elemento flotado a la izquierda.

both – Rompe la secuencia de flotados a ambos lados.

Column

Permite encolumnar el contenido de una caja.

column-count – Indica el número de columnas que deseamos generar.

column-width – Indica el ancho de cada columna.

column-gap – Espacio entre columnas.

column-rule-width, column-rule-style, column-rule-color – Estas propiedades tienen por finalidad establecer el ancho, estilo y color de la línea de separación entre columnas.

column-rule – Permite aplicar las tres reglas anteriores en una sola línea.

Position

Los elementos de una página HTML se colocan utilizando alguno de los siguientes posicionamientos:

- **Posicionamiento normal**: Cada elemento coge la posición según el orden de aparición.
- **Posicionamiento flotante**: Primero se coloca la caja en su posición normal (utilizando posicionamiento normal) y después se saca de esa posición para colocarlo lo más a la derecha (float: right) o lo más a la izquierda (float: left) posible.
- **Posicionamiento absoluto**: En el posicionamiento absoluto, el elemento es quitado de su normal flujo de entrada y posicionado con respecto al contenedor del que dependa.

Estos posicionamientos se gestionan mediante la propiedad position con los siguientes valores posibles:

static – Es el valor por defecto, se ubica siguiendo el flujo html

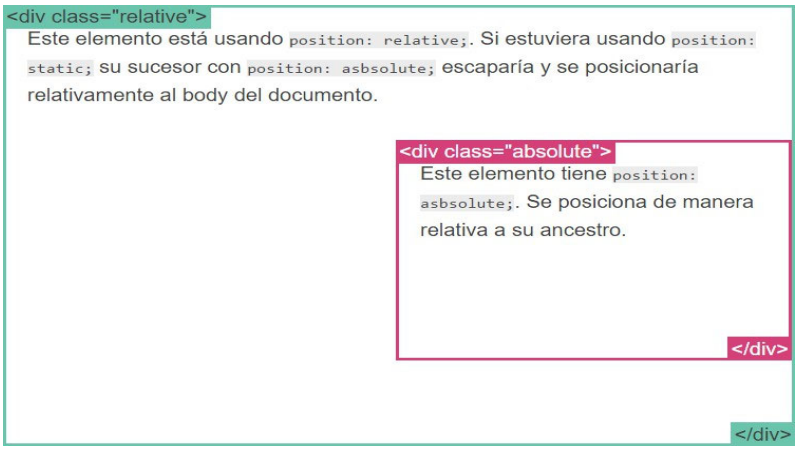
relative – Sigue el mismo flujo que static, pero se puede reajustar la posición utilizando las propiedades top, bottom, left, right.

absolute – Se comporta como el posicionamiento relativo, pero toma como referencia su ancestro relativo más cercano. Si no está contenido en ninguna caja con posicionamiento relativo toma como referencia el elemento body.

Un elemento absoluto debe estar dentro de un elemento posicionado (distinto de static).

fixed – saca al elemento del flujo otorgándole una posición fija dentro de la página, de forma que al hacer scroll el elemento permanece en el mismo sitio.

Sticky – Básicamente hace que el elemento se comporte como en position:relative hasta que, debido al scroll de la página, el elemento es alcanzado por la parte superior del viewport. En ese momento se comporta como fixed, y para que funcione correctamente hay que especificar al menos un valor top, bottom, left o right, en función de si va a ser sticky en un scroll horizontal o vertical.



Display

Esta propiedad permite redefinir como será mostrado un elemento. Los valores más utilizados son:

none – el elemento no se muestra y libera el espacio ocupado (desaparece).

inline – convierte un elemento de bloque en elemento de línea.

block – convierte un elemento de línea en elemento de bloque.

inline-block – muestra un contenedor de bloque en línea. El interior se comporta como elemento de bloque, el posicionamiento es como un elemento en línea.

table – deja que un elemento se comporte como un elemento de tabla. Otras propiedades usadas para esta representación en tabla:

- **table-caption** - el elemento se comporte como un título de tabla
- **Table-column-group** – simula un elemento <colgroup>
- **Table-header-group** – define una cabecera de tabla <thead>
- **Table-footer-group** - define un pie de tabla <tfoot>
- **Table-row-group** - el elemento se comporta como un elemento <tbody>
- **Table-cell** - Permite que el elemento se comporte como un elemento <td>
- **Table-column** - el elemento se comporta como un elemento <col>
- **Table-row** – El elemento se comporta como una fila <tr>

flex – Muestra un elemento como un contenedor flexible de nivel de bloque.

inline-flex – Muestra un elemento como un contenedor flex en línea.

grid – Muestra el contenedor como un conjunto de cuadrículas. Bloque.

inline-grid – Muestra el contenedor como un conjunto de cuadrículas. Línea.

Display: flex

Hasta la llegada de CSS3 se ha maquetado con técnicas como float o display table aunque no estaban pensados para utilizarse como los utilizamos. Flexbox sí es un estándar pensado para hacer layouts y por tanto soluciona la mayoría de las necesidades de los desarrolladores sin tener que emplear técnicas rebuscadas.

Display flex consta de 2 tipos de elementos, el contenedor y los ítems que van dentro del contenedor. Por lo que nos encontraremos con una serie de propiedades que afectan al contenedor y otras que afectan al contenido.

Supongamos que disponemos del siguiente código HTML

```
<section>
    <article>1</article>
    <article>2</article>
    <article>3</article>
    <article>4</article>
</section>
```

Al tratarse de elementos de bloque, después cada artículo se producirá un salto de línea.

Con el siguiente código CSS

```
section {
        display: flex;
}
```

Conseguiremos que todos los artículos se muestren en la misma línea.

Propiedades para el contenedor

flex-direction – Dirección del flujo de los elementos

- **row**: se colocan en fila de izquierda a derecha
- **row-reverse**: se colocan en fila, ero de derecha a izquierda
- **column**: se colocan uno debajo del otro
- **column-reverse**: se colocan uno debajo del otro, pero el último es el primero en aparecer.

flex-wrap – Gestiona como ser realizarán los saltos de línea.

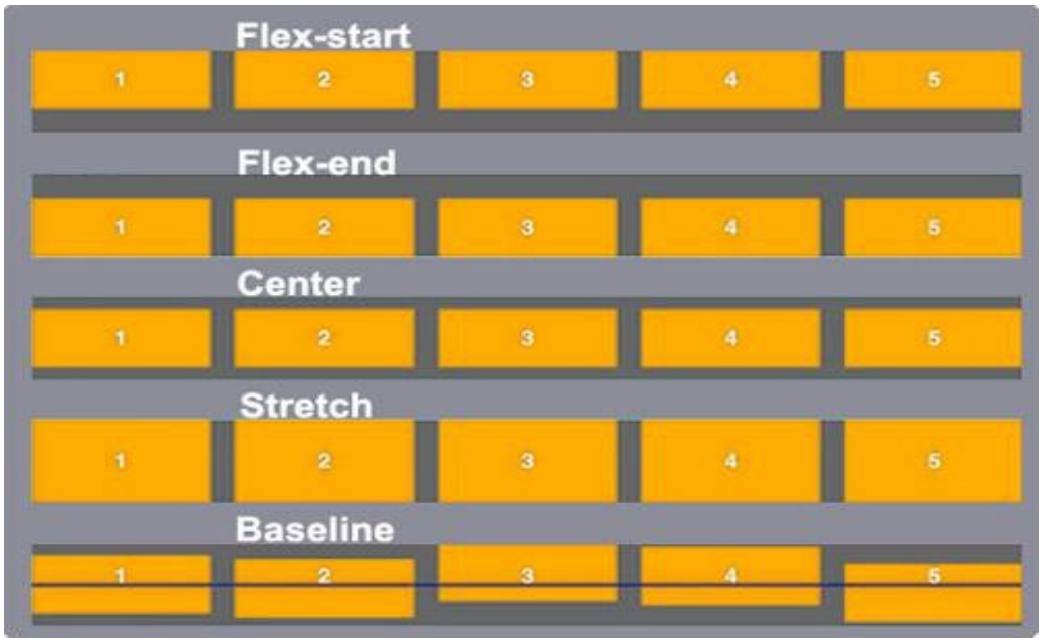
- **no-wrap**: no realiza saltos de línea reduciendo en ancho de los ítems. Este es el valor por defecto.
- **wrap**: si los elementos no entran en una línea se produce un salto.
- **wrap-reverse**: el salto de línea se produce hacia arriba.

justify-content – alineamiento y justificación de márgenes. Los valores más utilizados son:

- **flex-start** – coloca los elementos partiendo del inicio.
- **flex-end** – coloca los elementos partiendo del final.
- **center** – centra los elementos.
- **space-between** – reparte el espacio libre proporcionalmente el espacio entre los ítems, colocando el primero y el último pegados a los bordes.
- **space-around** – como el anterior hace un reparto proporcional, incluyendo al primero y al último.
- **space-evenly** – similar a **space-around** pero con un espaciado perfectamente uniforme entre y alrededor de cada elemento.

align-items – alineamiento y justificación verticales. Los valores más utilizados son:

- **flex-start** – alineados a la parte superior.
- **flex-end** – alineados a la parte inferior.
- **center** – centra los elementos, dejando el mismo espacio arriba y abajo.
- **strech** – ocuparán la altura total, si no se le ha marcado otra diferente.
- **baseline** – posicionará los elementos en función del texto que contengan.



Estos elementos se explicaron con los elementos posicionados horizontalmente, en caso de que estuvieran en vertical, sería lo mismo pero aplicado a lo ancho.

Propiedades para los Items

flex-grow – mediante un número especifica cuánto debe crecer nuestros flexs al rellenar el espacio sobrante.

flex-shrink – Al contrario que la propiedad anterior, especifica el comportamiento de los flexs cuando el tamaño del contenedor es menor al de los flex que lo componen.

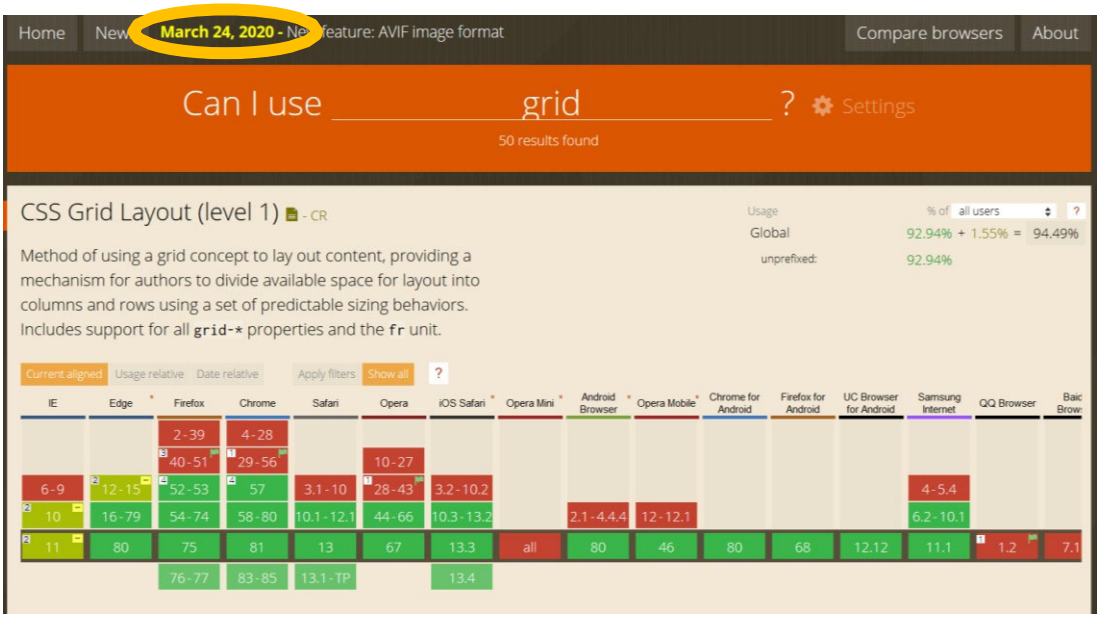
flex-basis – indica un punto de partida desde el cual el navegador se guiaría para calcular el tamaño del flex (aunque como dependiendo de las otras propiedades, este tamaño podría variar).

order – especifica el orden de los elementos flexibles, con lo que un elemento que está en la posición 3 se puede pasar a la 1.

Display: grid

El módulo grid CSS ofrece un sistema de diseño basado en cuadrícula, con filas y columnas, lo que facilita el diseño de páginas web sin tener que usar float y position. Grid Layout consta de un elemento padre y uno o más elementos secundarios o hijos que se distribuyen en filas y columnas.

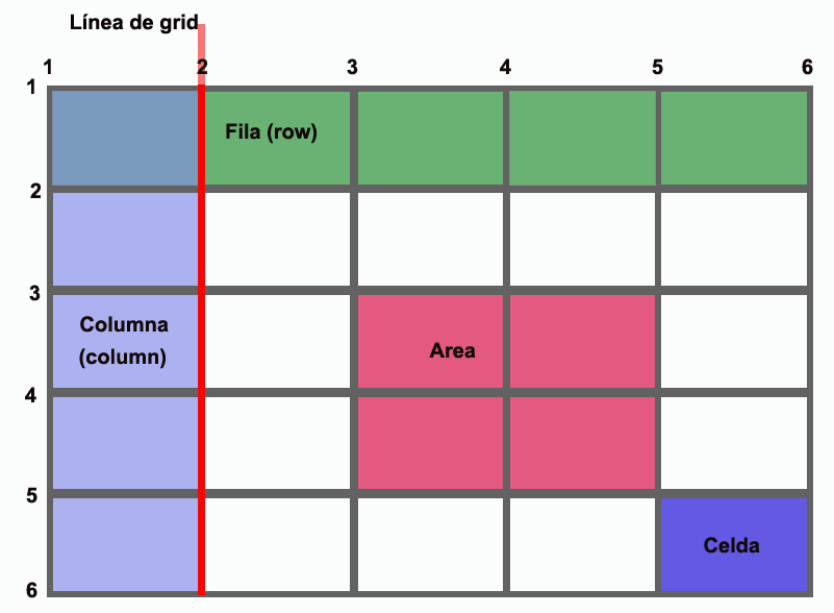
Grid es compatible con todos los navegadores modernos.



Con Flexbox sólo se podía definir qué hacer con los items en el eje horizontal o en el eje vertical. Por ejemplo, cuando en el eje horizontal se termina el espacio, los items se iban colocando en la siguiente fila, aunque no se tenía el control de las dos filas.

La diferencia es que en grid hay realmente dos dimensiones. Se podrá decidir qué celdas ocupará cada elemento, controlando perfectamente las dimensiones de cada fila o columna.

Elementos del grid



Líneas de grid

Un grid se parece a una matriz, donde las líneas del grid enmarcan cada **celda**. Por lo tanto, si el grid tiene 5 columnas, tendrá 6 líneas verticales, y si tiene 4 filas, tendrá 5 líneas horizontales. Hay que destacar que en este caso se empieza a contar desde el 1.

Columnas y filas

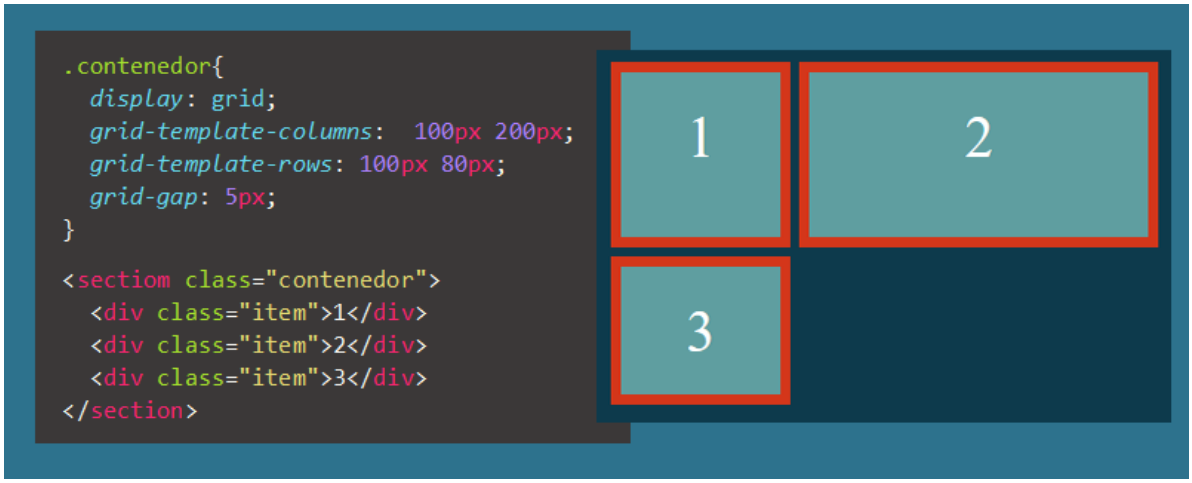
También llamadas **track**. Una fila (horizontal) irá de izquierda a derecha, ocupando una única línea completa y un track vertical o columna, va de arriba abajo.

Área

Es un grupo de celdas que forman un cuadrado o rectángulo.

Definición del grid

```
.contenedor {
        display: grid;
}
```



grid-template-columns

- Permite definir la cantidad y el tamaño de las columnas.
- Se pueden usar unidades fijas o relativas.
- Aparece un nuevo tipo de unidad de medida **fr** (fraction units). 1fr equivale al 100% del espacio restante. (10% 1fr) (1fr 2fr 1fr) (1fr 1fr 1fr 1fr).
- Esta última distribución puede ser muy 'cansina', por ello se puede utilizar la función **repeat**(4, 1fr), que permite repetir 4 veces 1fr. **repeat**(2, 10% 1fr), haría 4 columnas (10% 1fr 10% 1fr).
- Es posible establecer un tamaño mínimo o máximo de una fila o columna con **minmax**(100px, 200px), pudiendo combinarlo con los otros formatos, por ejemplo (minmax (100px, 10%) 1fr 2fr); **repeat** (**minmax** (100px, 10%) 1fr);

grid-template-rows:

Para la definición de las filas, se trata de una definición explícita de filas y/o columnas.

A diferencia delo que sucede con las columnas, aquí el uso de **fr** y **%** es más complicado al no conocerse la altura total del contenedor. Una solución es utilizar **vh** para marcar esa altura.

grid-gap:

Define la separación entre celdas, tanto entre filas como entre columnas. Si se dan 2 valores, el primero se refiere a la separación entre filas y el 2 entre columnas.

También se pueden gestionar separadamente con **grid-row-gap** y **grid-column-gap**.

```
.contenedor {
        grid-gap: 20px 10px;
}
```

grid-auto-flow:

Cuando el flujo sobrepasa el contenedor (no caben todos los elementos), se indica donde se deben a colocar.

Los valores permitidos son:

- grid-auto-flow: row; es el valor por defecto, el excedente lo pasa a las filas siguientes.
- grid-auto-flow: column; pone todos los elementos en la misma fila (aparece barra de scroll).
- grid-auto-flow: dense; intenta rellenar huecos con elementos más pequeños que aparezcan más tarde. Esto puede hacer que los elementos aparezcan fuera de orden.
- grid-auto-flow: row dense;
- grid-auto-flow: column dense;

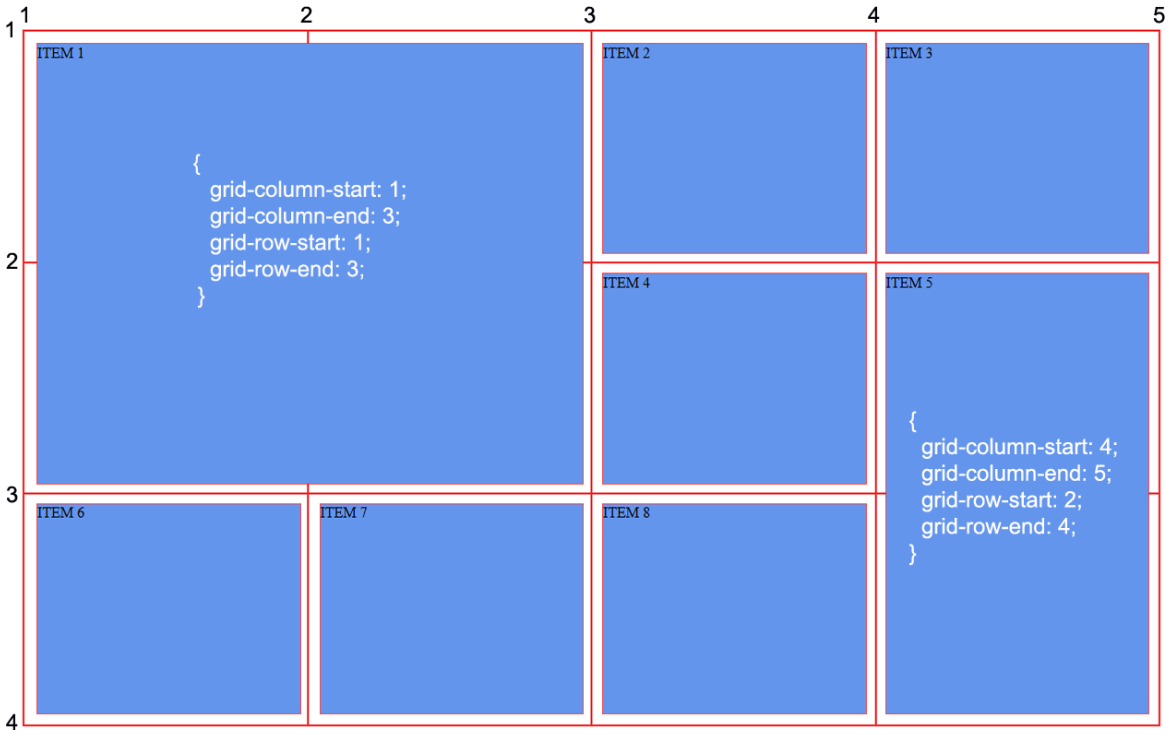
grid-auto-columns, grid-auto-rows

Permite indicar el ancho de las columnas y filas extra, aunque el caso de las filas no es muy útil.

Definición de áreas

Las propiedades **grid-column** y **grid-row** permiten definir las áreas o zonas en los que se posicionará un elemento, incluso superponiendo unas celdas con otras. Cada una de estas propiedades recibirán dos valores **start** y **end**. Además, se pueden definir de otras formas que se verán a continuación.

```
.contenedor {
        height: 90vh;                    // altura del viewport (pantalla)
        display: grid;
        grid-template-columns: repeat(4, 1fr);
        grid-template-rows: repeat(3, 1fr);
        grid-gap: 20px;
}
```



La imagen anterior muestra una de las soluciones, indicando star y end en cada elemento.

Esto también se podría haber resuelto de esta forma:

```
.item1{
        grid-column : 1 / 3;          //shorthand o abreviatura
        grid-row : 1 / 3;
}
.item5{
        grid-column : 4 / 5;          grid-column : 4 / -1;
        grid-row : 2 / 4;
}
```

grid-column: 1 / 3

Hace que la columna del elemento indicado (**item1**) empiece en la línea 1 y acabe en la 3, recordar que si hay 4 columnas hay 5 líneas, ya que se cuentan los "bordes" o líneas de las columnas empezando en el 1 (lo mismo para las filas).

grid-column: 4 / -1

En el caso del **item5** podríamos hacerlo de ambas formas, con la segunda (4 / -1) se le indica que vaya hasta la última posición horizontal del grid. Con -2 se le indicaría la penúltima, y así sucesivamente.

Debemos pensar en las columnas como un elemento circular, en el que podemos movernos hacia adelante o hacia atrás, por lo que la línea anterior a la primera además de ser la última es la -1, y -2 sería la penúltima.

grid-column: span X; grid-row: span X;

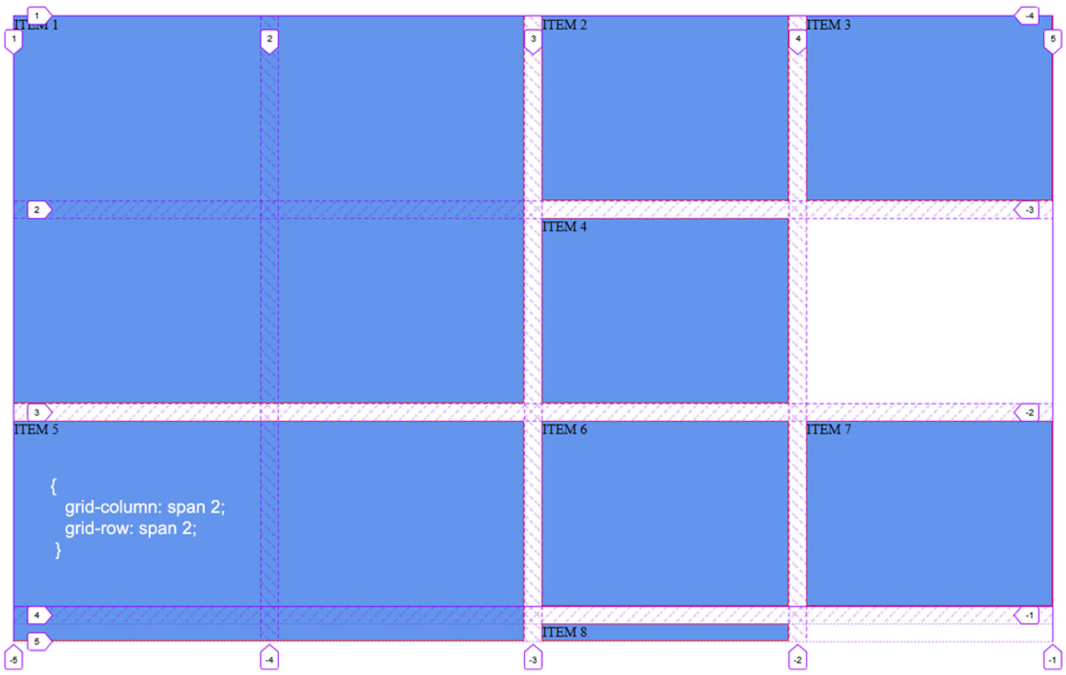
Se asemejan al funcionamiento del colspan y rowspan de las tablas. Permiten indicar cuanto se extiende una columna o fila.

Siguiendo con el anterior ejemplo, ahora lo podríamos solucionar de la siguiente forma:

```
.item1{
        grid-column : span 2;
        grid-row : span 2;
}
.item5{
        grid-column : span 1;          // no haría falta;
        grid-row : span 2;
}
```

Si el ítem no tiene suficiente espacio para crecer, se desplaza dentro del grid hasta encontrarlo.

El inspector de elementos del navegador (en este caso Firefox Developer Edition) permite activar la opción de grid o rejilla para poder ver las líneas y la distribución del grid, permitiendo ver el desplazamiento del ITEM 5.



Poner nombres a las líneas

En la definición del grid, al hacer la distribución de las columnas, podemos agregar nombres a las líneas:

```
.contenedor{
        display : grid;
        grid-template-columns: [content-start] 8fr [content-end side-start] 2fr [side-end]
}
```

La anterior definición indicaría que se divide el grid en 2 columnas:

- la primera del 80% que empieza en *content-start* y acaba en *content-end* (estos nombres serían a gusto del desarrollador).
- la segunda del 20% empieza en *side-start* y acaba en *side-end.*

Ahora la primera línea tiene dos formas de ser llamada, *1* ó *content-start*, la segunda pasa a tener 3 nombres, *2*, *content-end* y *side-start.*

Ahora podríamos hacer la siguiente distribución:

```
section {
        grid-column : content-start / content-end;
}
aside {
        grid-column : side-start / side-end;
}
```

Nombres áreas

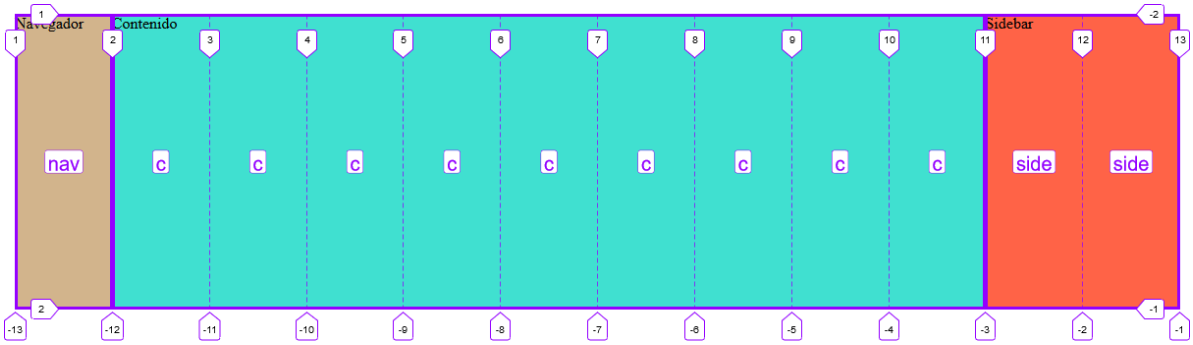
Esta sería otra solución a la definición de un grid, en este caso nos referiríamos a las zonas de grid, en lugar de referenciarlo con números.

```
.contenedor {
        grid-template-columns:  repeat(12, 1fr);
        grid-template-areas : "nav c c c c c c c c c side side";
}
```

Nuestro grid tendría una columna para el nav (navegador), 9 para c (contenido) y 2 para side (sidebar).

```
nav{
        grid-area : nav;
}
section{
        grid-area : c;
}
aside{
        grid-area : side;
}
```

El inspector de elementos también nos puede mostrar los nombres de las áreas



Si queremos dejar una celda o columna vacía, sustituyes el elemento por un punto “.”

```
.contenedor {
        grid-template-columns:  150px 1fr;
        grid-template-rows: 100px 1fr 80px;
        grid-template-areas : "header header"
                "side  contenido"
                "footer  footer";
}
header{
        grid-area : header;
        background-color: tan;
}
aside{
        grid-area : side;
        background-color: tomato;
}
section{
        grid-area : contenido;
        background-color: turquoise;
}
footer{
        grid-area : footer;
        background-color: thistle;
}
```

Order

Similar al uso en flexbox. Permite cambiar el orden de aparición de los elementos cambiando el flujo normal sin tener que tocar el HTML.

El valor más bajo se presenta más arriba (o antes), en caso de 2 elementos en el mismo order, se posicionan en orden de aparición.

Transiciones

Proporcionan una forma de animar los cambios de las propiedades CSS, en lugar de que los cambios surtan efecto de manera instantánea.

transition-property - Especifica el nombre o nombres de las propiedades CSS a las que deberían aplicarse las transiciones. Sólo las propiedades que se enumeran aquí son animadas durante las transiciones; los cambios en el resto de las propiedades suceden de manera instantánea como siempre. Si en lugar de alguna propiedad utilizamos la palabra reservada **ALL**, animarían todas al mismo tiempo.

> background-color, border, border-radius, color, top, bottom, left, right, box-shadow, clip, fill, height, width, line-height, letter-spacing, margin, opacity, outline, stroke, text-shadow, vertical-align, word-spacing, visibility, z-index.

transition-duration - Especifica la duración en la que sucederán las transiciones. Puedes especificar una única duración que se aplique a todas las propiedades durante la transición o valores múltiples que permitan a cada propiedad de transición un período de tiempo diferente. Hay que poner el nombre de las unidades.

transition-delay - Define el tiempo de espera entre el momento en que se cambia una propiedad y el inicio de la transición.

transition-timing-function - Especifica la curva cúbica bézier que se usa para definir cómo se computan los valores intermedios para las propiedades. Existen valores de intervalos predeterminados:

- **linear**.- La animación se realiza de manera uniforme
- **ease**.- La animación acelera al inicio se retarda un poco y se acelera al final de nuevo
- **ease-in**.- La animación se retarda al inicio, pero lo repone al final
- **ease-out**.-La animación se acelera al inicio pero se retarda al final,
- **ease-in-out**.- La animación se retarda al inicio se acelera un poco luego se retarda al final de nuevo.

Transformaciones

Transform es la propiedad que permite aplicar transformaciones 2D o 3D a un elemento, por ejemplo: rotar, escalar, mover . . .

rotate - Define una operación de rotación 2D de un elemento, especificando la cantidad de grados (deg) que este rotará en sentido de las manecillas del reloj (según lo especificado por la propiedad transform-origen).

```
transform:  rotate(deg);                    /* ej. rotate(90deg) */
```

rotateX, rotateY - Define una operación de rotación 3D de un elemento en el eje X o Y respectivamente

```
transform:  rotateX(deg);                   /* ej. rotateX(90deg) */
```

rotateZ - Define una operación de rotación 3D de un elemento en el eje Z.

```
transform:  rotateZ(deg);                   /* ej. rotateZ(90deg) */
```

scale - Especifica una operación de escalado 2D descrita por [sx, sy], si se omite el valor sy, se asumirá que tanto sx como sy tendrán el mismo valor

```
transform:  scale(sx[, sy]);                /* ej. scale(2.5, 4)*/
```

scaleX, scaleY - Especifica una operación de escalado 2D usando el vector [sx, 1] o [1, sy].

```
transform:  scaleX(sx);                     /* ej. scale(2.5)*/
```

skewX, skewY - Sesga un elemento a lo largo del eje X o Y por el ángulo dado.

```
transform:  skewY(deg);                     /* ej. skew(180deg)*/
```

translate, translateX, translateY - Especifica una translación 2D dada por el vector [tx, ty]. Si ty no es especificada, se asumirá que su valor es cero. Cada translation-value puede ser un valor de longitud o un valor de porcentaje.

```
transform:  translate(tx[, ty]);          /* ej. translate(50px, 100px) */
```

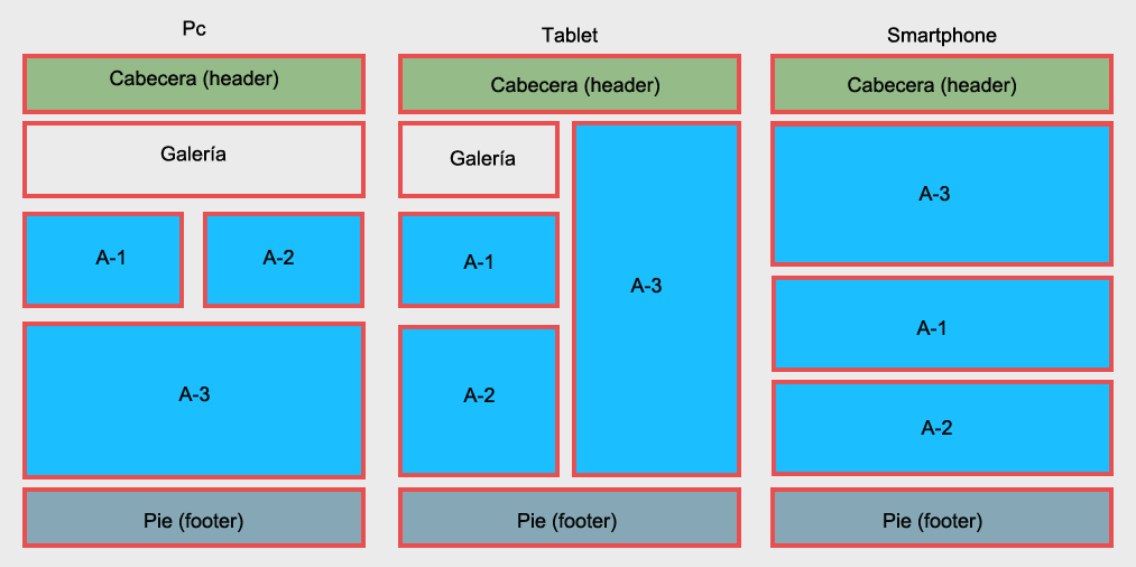
```
transform:  translateY(ty);               /* ej. translateY(100px) */
```

Responsive web design

Tanto el diseño responsive como adaptativo se basa en :

- grids fluidos
- imágenes flexibles
- consultas de medios (@media)

Pero lo primordial es planificar el diseño de nuestra web antes



Lo que buscan es adaptar la estructura de la web y todos los elementos que lo integran a la pantalla de nuestro dispositivo para ofrecer el mejor aspecto visual y una gran funcionalidad atendiendo a criterios como la accesibilidad a los contenidos.

Aunque diseño responsive y adaptativo no son exactamente lo mismo.

- **El diseño web de tipo responsive** responde al tamaño del navegador en cualquier punto dado.
    - se establecerán valores de tamaño proporcionales en lugar de establecer valores fijos, así se consigue que entre dos medidas el diseño se reposicionen los elementos.
- **El diseño web adaptativo** se adapta al ancho del navegador en puntos específicos. En otras palabras, al sitio web solo le preocupa que el navegador tenga un ancho específico.
    - tamaños de pantalla fijos y preestablecidos (media queries) para cada uno de los dispositivos, se fija un tamaño del contenedor principal que se mantiene para todos los dispositivos que se encuentren en ese rango.

En un primer paso se añadirá una etiqueta meta en la cabecera del documento.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

A continuación usaremos las media queries para definir los distintos tamaños de los dispositivos, y dentro de ellas especificaremos los cambios necesarios.

```
@media (max-width: 991.98px) {
        body {
                background-color: grey;
                color: white;
        }
}
```

Hay 2 formas de enfrentarse al diseño

1. A partir del diseño más grande (el de escritorio)

//Pantallas muy grandes (desktops de más de 1200px de ancho)

```
//No hace falta media-query porque será nuestro diseño por defecto
```

//Pantallas grandes (desktops de menos de 1200px)

```
@media (max-width: 1199.98px) { ... }
```

//Pantallas medianas (tablets de menos de 992px)

```
@media (max-width: 991.98px) { ... }
```

//Pantallas pequeñas (móviles en landscape de menos de 768px)

```
@media (max-width: 767.98px) { ... }
```

//Pantallas muy pequeñas (móviles en portrait de menos de 576px)

```
@media (max-width: 575.98px) { ... }
```

2. A partir del más pequeño (el de móvil)

//Pantallas muy pequeñas (móviles en portrait de menos de 576px)

```
//No hace falta media-query porque será nuestro diseño por defecto
```

//Pantallas pequeñas (móviles en landscape de más de 576px)

```
@media (min-width: 576px) { ... }
```

//Pantallas medianas (tablets de más de 768px)

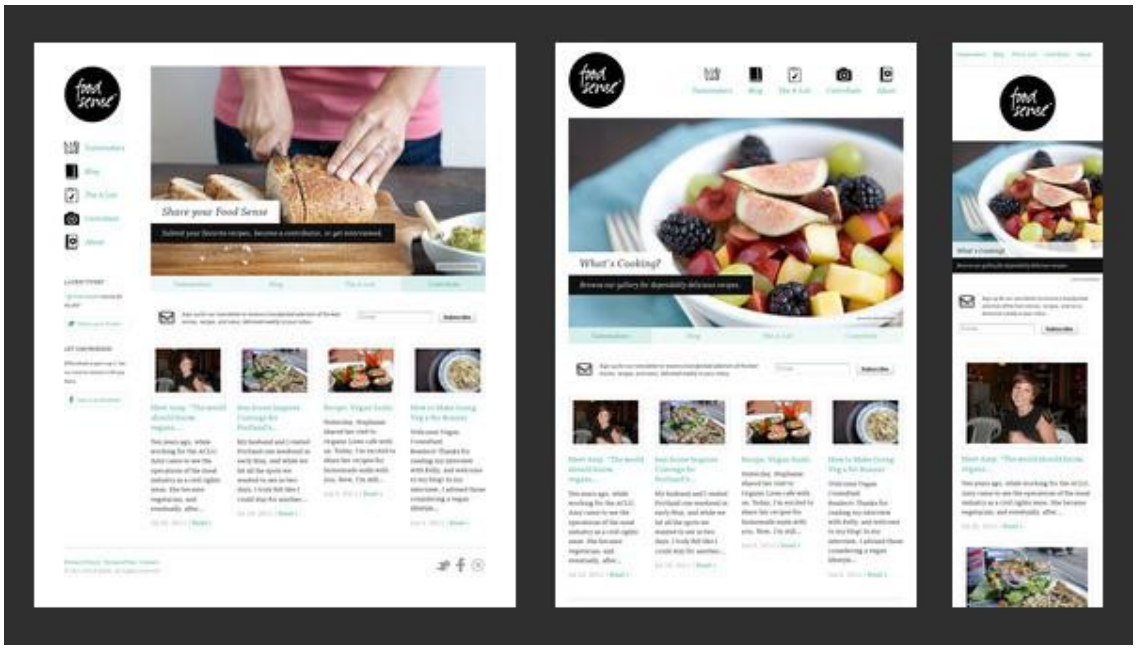
```
@media (min-width: 768px) { ... }
```

//Pantallas grandes (desktops de más de 992px)

```
@media (min-width: 992px) { ... }
```

//Pantallas muy grandes (desktops de más de 1200px)

```
@media (min-width: 1200px) { ... }
```




INQUIRER
"Give me problems, give me work."
victors & villains
INQUIRER
"Give me problems, give me work."
victors & villains
INQUIRER

Contenido