

UD2. Incluir distinto tipo de contenidos en una web con HTML

Contenido

- 1. Introducción
- 2. Versiones
- 3. Etiquetas HTML
 - 3.1. Formato
 - 3.2. Atributos
 - 3.3. Estructura del documento
 - 3.3.1. Cabecera (head)
 - 3.3.2. El cuerpo (body)
 - 3.3.3. El texto
 - 3.3.4. Tablas
 - 3.3.5. Imágenes
 - 3.3.6. Multimedia
 - 3.3.7. Enlaces
 - 3.3.8. Listas
 - 3.3.9. Formularios
 - 3.4 iframes
 - 3.5 Otros



1. Introducción

- El lenguaje de marcas **HTML** (Hyper Text Markup Language) surgió por la complejidad de SGML, creándose un lenguaje mucho más simple y adaptado expresamente a representar contenido para la web.
- El número de etiquetas del que se dotó a HTML era considerablemente reducido, lo que hacía que se curva de aprendizaje fuera bastante rápida.
- Un documento realizado con el lenguaje HTML consta de dos elementos: los contenidos del documento y las instrucciones HTML que darán el formato adecuado a dichos contenidos.
- Estas instrucciones, llamadas etiquetas, constituyen la base del lenguaje HTML.

2. Versiones

- **1993 HTML 1.0** Permitía textos e imágenes
- **1995.** HTML 2.0. Estructura de código más limpia y se estandariza normativa para los navegadores. El contenido de las páginas era más importante que el diseño.
- **1997.** HTML 3.2. Los navegadores soportan tags y frames. La primera versión desarrollada y estandarizada exclusivamente por el W3C. Integración de CSS.
- **1999.** HTML 4.0. Más estilos y scripts. Mejora de la accesibilidad de las páginas diseñadas, tablas complejas y mejoras en los formularios.
- **2014.** HTML 5.0 Nuevas etiquetas: email, password, audio, vídeo, canvas. En definitiva un código más simple y universal.

2. Versiones

- **2008.** HTML 5. Nuevas funcionalidades como arrastrar imágenes, mejoras en los formularios y facilidades para validar el contenido sin Javascript, permite generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido, etc.

HTML



2. Versiones

- Para que los navegadores sepan que versión de HTML es, el documento que debe presentar, es necesario incluir una cabecera DOCTYPE que lo identifique.
- A continuación se muestra un ejemplo para HTML 4.01 y su uso en HTML5:

HTML4:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

HTML5:

```
<!DOCTYPE html>
```

2. Versiones

- HTML5 no es una nueva versión del antiguo lenguaje de etiquetas, ni siquiera una mejora de esta ya antigua tecnología, sino un nuevo concepto para la construcción de sitios web y aplicaciones en una era que combina dispositivos móviles, computación en la nube y trabajos en red.
- Todo comenzó mucho tiempo atrás con una simple versión de HTML propuesta para crear la estructura básica de páginas web, organizar su contenido y compartir información.

2. Versiones



2. Versiones

World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#), [Policy](#), November's [W3 news](#), [Frequently Asked Questions](#).

[What's out there?](#)

Pointers to the world's online information, [subjects](#), [W3 servers](#), etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#), [X11 Viola](#), [NeXTStep](#), [Servers](#), [Tools](#), [Mail robot](#), [Library](#))

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

A summary of the history of the project.

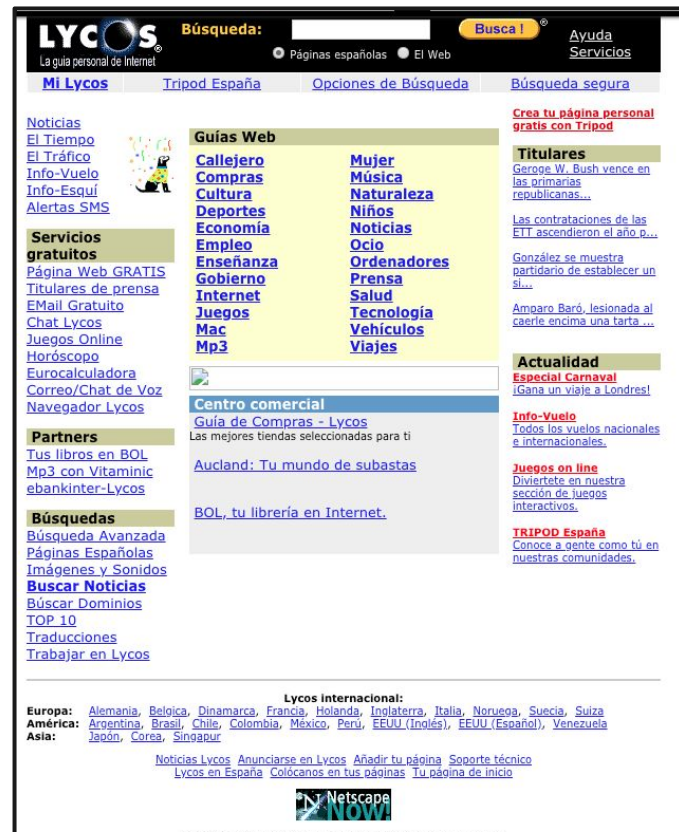
[How can I help?](#)

If you would like to support the web..

[Gettting code](#)

Getting the code by [anonymous FTP](#), etc.

1990: Primera web publicada en Internet
[WorldWideWeb](#)



2000: Buscador Lycos
[Lycos](#)

3. Etiquetas HTML

3.1. Formato

- Una **etiqueta** o marca está formada por una o varias palabras reservadas, es decir, palabras que tienen un significado especial en el lenguaje.
- Se diferencian de la información de la información del documento propiamente dicha porque se encuentran encerradas entre los símbolos "<" y ">"
- Existen dos tipos básicos de etiquetas:
 - **Las de inicio o apertura:** declaran las características o formatos de elemento (por ejemplo, indica que el texto irá en negrita o cursiva).
 - **Las de fin o cierre:** indica al navegador que el formato que afectaba al elemento desde la etiqueta de inicio, se acaba.
- Las etiquetas de apertura y cierre son exactamente iguales salvo por el símbolo "/"

3. Etiquetas HTML

3.1. Formato

- Ejemplo:
 - Si se inserta la etiqueta de “texto en negrita” (****), el navegador formateará en cursiva todas las palabras que vengan a continuación.
 - El efecto dejará de aplicarse cuando el navegador encuentre la etiqueta de cierre (**</>**)

**** Este texto se mostrará en negrita ****

3. Etiquetas HTML

3.1. Formato

- No todas las etiquetas de este lenguaje tienen su correspondiente etiqueta de fin, ya que hay diferentes tipos:
 - **Etiquetas de apertura y cierre.** La instrucción sólo se aplica al elemento que encierran
 - **Etiquetas de sólo apertura.** El efecto se produce en un punto determinado del documento sin afectar a otros elementos.

`<p> Este texto se mostrará en negrita </p>`

`
`

3. Etiquetas HTML

3.1. Formato

- Las etiquetas con apertura y cierre deben estar “balanceadas”, es decir, si se abren en un orden determinado, se deben cerrar en el orden inverso para que las primeras engloben a las siguientes.

```
<etiqueta 1>  
    <etiqueta 2>  
  
        </etiqueta 2>  
    </etiqueta 1>
```

3. Etiquetas HTML

3.1. Formato

- Por otra parte, HTML no diferencia entre mayúsculas y minúsculas a la hora de escribir e interpretar etiquetas.
- Con el fin de lograr una mayor uniformidad en el código, lo normal es que siempre se escriban de la misma forma, prefiriendo la mayoría de los desarrolladores ponerlas en minúsculas siguiendo las recomendaciones del W3C.

3. Etiquetas HTML

3.2. Atributos

- Los **atributos** modifican el comportamiento general de la etiqueta, ya sea cambiando el color, alineación, estilos, etc.
- Generalmente, están formados por el nombre del atributo, que es una palabra reservada del lenguaje, el signo “=” seguido de comillas y el valor que toma.
- Una etiqueta genérica con atributos sería:

<etiqueta atributo1=“valor1” atributo2=“valor2”> elemento </etiqueta>

3. Etiquetas HTML

3.2. Atributos

- El orden en el que se incluyan los atributos es indiferente, aunque hay que tener en cuenta:
 - Si el valor del atributo contiene dos o más palabras separadas por espacios, debe ir entre comillas para evitar que el navegador malinterprete el código.
 - Cada atributo sólo puede tener un valor en cada etiqueta.

3. Etiquetas HTML

3.3. Estructura del documento

- Un documento realizado en HTML tiene unas partes bien definidas que constituyen su estructura básica o lo que se conoce como el esqueleto del documento.
- Estas partes están definidas por marcas o etiquetas.
- Para que el navegador interprete correctamente la página Web en cuestión, debe saber el lenguaje en el que está escrita.
- Esto se le indica de dos formas:
 - en primer lugar, el documento llevará la extensión html, aunque también admite htm;
 - en segundo lugar, el documento llevará al principio y al final las etiquetas <html> y </html>, las cuales indican el lenguaje en el que está escrito el mismo.

3. Etiquetas HTML

3.3. Estructura del documento

HTML4:

```
<html>
```

XHTML:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
```

HTML5:

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

3. Etiquetas HTML

3.3. Estructura del documento

Un sitio web está compuesto por distintas páginas web.

Los archivos HTML incluirán los contenidos usando los distintos **elementos** disponibles:

- Textos
- Enlaces
- Listados
- Tablas
- Multimedia: imágenes, vídeo y audio
- Marcos para mostrar en un recuadro otro sitio web
- Formularios

Los **formatos, maquetación y flexibilidad** de las páginas se establecerán con lenguaje **CSS**

3. Etiquetas HTML

3.3. Estructura del documento

- Una vez establecido el lenguaje que se utiliza, se escribe el esqueleto del documento.
- Éste está constituido por dos partes:
 - Cabecera: está delimitada por las etiquetas <head> </head>.
 - Cuerpo: está delimitado por <body> </body>.
- De esta forma, la estructura genérica en HTML será

```
<html>  
  <head>  
    Elementos de la cabecera  
  </head>  
  <body>  
    Contenido del documento  
  </body>  
</html>
```



Las etiquetas head y body son opcionales aunque se recomienda introducirlas para identificar mejor las partes del documento y estructurar el código.
En caso de que un documento HTML no presente las etiquetas identificativas, el navegador considerará que todo lo que se defina en el fichero forma parte del cuerpo del documento.

3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera

- En la cabecera se incluirán las definiciones generales que afectan a todo el documento.
- Todas son **opcionales** y se utilizan en casos muy concretos.



```
<head>
<meta name="TITLE" content="..."
<meta name="KEYWORDS" content="..."
<meta name="DESCRIPTION" content="..."
<link rel="stylesheet" type="text/css" href="..."
<script language="javascript" src="..."
</head>
<body bgcolor="#ffffff" >
```

3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera

- La etiqueta **<meta>**
 - Se utiliza para añadir información sobre la página.
 - Esta información puede ser utilizada por los buscadores.
 - El atributo **name** indica el tipo de información, y el atributo **content** indica el valor de dicha información.

| name | uso |
|----------------|---|
| author | Indicar quien confeccionó la página web |
| description | Resumen del contenido de la página |
| keywords | Palabras clave (similar a los # en RRSS) |
| classification | Palabras que ayudan al buscador a clasificar el sitio |
| generator | Indicar el programa usado para generar la página |

3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera

- La etiqueta **<meta>**
 - No necesita etiqueta de cierre.
 - Para cada etiqueta **<meta>** solo es posible indicar un tipo de información y su valor, pero es posible insertar varias etiquetas **<meta>** en un mismo documento.

```
<html>
  <head>
    ...
    <meta name="author" content="Tim Berners-Lee" />
    <meta name="description" content="HTML básico" />
    <meta name="keywords" content="código, HTML, etiqueta, web" />
  </head>
```

3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera

- La etiqueta **<meta>**
 - También se utiliza para indicarle al navegador qué tipo de caracteres va a encontrarse y así que muestre correctamente un acento o una ñ por ejemplo.
 - El valor para el atributo charset suele ser “UTF-8” o UNICODE

```
<html>
  <head>
    <meta charset="UTF-8"/>
  </head>
```

- Existe una variante para indicar lo mismo

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
```


3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera

- La etiqueta **<meta>**
 - También se utiliza para indicarle al navegador alguna información o alguna acción que debe realizar.
 - En este caso se utiliza el atributo http-equiv, en lugar de name.
 - Por ejemplo, imaginemos que queremos que nuestra página se actualice automáticamente cada 30 segundos.

```
<html>
  <head>
    ...
    <meta http-equiv="Refresh" content="30" />
  </head>
```

3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera

- La etiqueta **<meta>**
 - Ahora imaginemos que hemos cambiado la dirección en la que se encuentra nuestra página web, y queremos que cuando algún usuario visite la página en la URL antigua, a los 5 seg. el navegador lo redirija automáticamente a la URL nueva.

```
<html>
  <head>
    ...
    <meta http-equiv="Refresh" content="5; URL=http://nuevaURL.com" />
  </head>
```

3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera

- La etiqueta **<meta>**
 - Meta keywords y meta description para SEO: ¿funcionan todavía?
 - Este artículo describe los resultados de un experimento que intenta comprobar con hechos si Google tiene en cuenta o no las meta keywords y la meta description a la hora de posicionar una web en los resultados.



SEO (Search Engine Optimization)

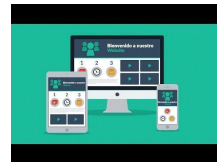
El SEO es el proceso de optimizar el contenido y la estructura de un sitio web para aumentar su visibilidad en los resultados orgánicos de los diferentes motores de búsqueda.

3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera

Diseño responsive



- La etiqueta **<meta>**
 - HTML5 introdujo un método para que los diseñadores web tienen control sobre el área de visualización. El visor es zona visible del usuario de una página web.
 - La anchura = dispositivo de ancho de pieza establece el ancho de la página para seguir la pantalla-anchura del dispositivo (que variará en función del dispositivo).
 - El inicial escala = 1.0 parte establece el nivel de zoom inicial cuando la página se carga primero por el navegador.

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

3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera



Sin la etiqueta de la ventana gráfica meta



Con la etiqueta de la ventana gráfica meta

3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera

- La etiqueta **<title>**
 - Esta etiqueta nos permite establecer el título que tendrá nuestra página, este es mostrado en la barra de título de nuestro navegador o en la pestaña correspondiente

```
<html>
  <head>
    <title> Mi primer proyecto web</title>
  </head>
```

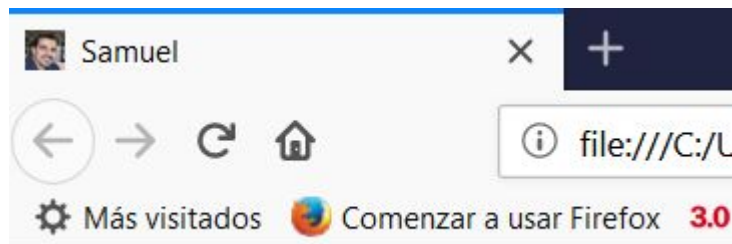
3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera

Puedes insertar un icono en la pestaña de tu web con el siguiente comando:

```
<html>
  <head>
    <link rel="shortcut icon" href="imagen.png" />
  </head>
```



3. Etiquetas HTML

3.3. Estructura del documento

La Cabecera

La etiqueta link también nos servirá para hacer una llamada a una hoja de estilos externa CSS.

```
<html>  
  <head>  
    <link rel="stylesheet" href="stilos.css"/>  
  </head>
```


3. Etiquetas HTML

3.3. Estructura del documento

El Cuerpo <body>

- El cuerpo del documento viene indicado por el par de etiquetas **<body> </body>**.
- Es en él donde van dispuestos los distintos elementos que componen el documento y que le dan sentido: texto, imágenes, sonido, etc.
- La etiqueta <body> admite una serie de atributos que son de carácter global para todo el documento.
- Es decir, define aspectos generales del documento como la imagen de fondo de la página o el color de la misma, etc.

3. Etiquetas HTML

3.3. Estructura del documento

El texto en HTML

- **La etiqueta <p>**

- La etiqueta <p> </p> o de párrafo, es una de las más utilizadas en HTML a la hora de estructurar un texto.
- Introduce un espacio de dos líneas de separación (equivalente a pulsar dos veces Enter en un editor de texto) con el siguiente párrafo de texto.
- Su uso es muy recomendable al servir como separador de bloques de texto y elemento de espaciado.

<p> Párrafo </p>

3. Etiquetas HTML

3.3. Estructura del documento

El texto en HTML

- **Títulos, la etiqueta <h...>**

En cualquier documento es necesario diferenciar los diferentes apartados mediante el uso de encabezados o títulos, HTML nos permite crear títulos mediante la etiqueta <h1> hasta <h6> definiendo cada una de ellas un tamaño de letra. Además servirá a los buscadores para conocer el contenido estructurado de la página.

```
<body>
  <h1>Titulo página</h1>
  <h2>Titulo apartado</h2>
  <h3>Titulo subapartado</h3>
  <h4>Titulos secundarios</h4>
  <h5>Titulos secundarios</h5>
  <h6>Titulos secundarios</h6>
</body>
```

Salida por
navegador

Titulo página

Titulo apartado

Titulo subapartado

Titulos secundarios

Titulos secundarios

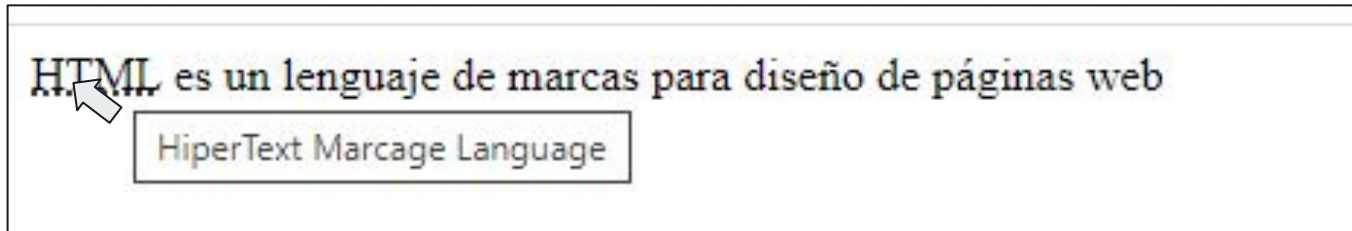
Titulos secundarios

3. Etiquetas HTML

3.3. Estructura del documento

- Etiqueta <abbr>

```
<p><abbr title="HiperText Marcage Language">HTML</abbr>  
es un lenguaje de marcas para diseño de páginas web</p>
```



3. Etiquetas HTML

3.3. Estructura del documento

El texto en HTML

- La etiqueta `
`
 - Introduce un salto de línea en nuestro documento

```
Line1 <br/> Line2
```



3. Etiquetas HTML

3.3. Estructura del documento

El texto en HTML


- **La etiqueta <hr/>**
 - Introduce una línea horizontal en el documento.
 - Por defecto, dicha línea poseerá el ancho de la ventana del navegador, tendrá forma 3D e introducirá una separación equivalente a un cambio de párrafo tanto por delante como por detrás de ella.

3. Etiquetas HTML

3.3. Estructura del documento

- Existen otras muchas etiquetas que nos permiten trabajar con texto dentro de HTML

W3C HTML Element Reference

| Tag | Description |
|---|---|
| <u><acronym></u> | Not supported in HTML5. Use <abbr> instead. Defines an acronym |
| <u><abbr></u> | Defines an abbreviation or an acronym |
| <u><address></u> | Defines contact information for the author/owner of a document/article |
| <u></u> | Defines bold text |
| <u><bdi></u> |  Isolates a part of text that might be formatted in a different direction from other text outside it |
| <u><bdo></u> | Overrides the current text direction |
| <u><big></u> | Not supported in HTML5. Use CSS instead. Defines big text |
| <u><blockquote></u> | Defines a section that is quoted from another source |
| <u><center></u> | Not supported in HTML5. Use CSS instead. Defines centered text |
| <u><cite></u> | Defines the title of a work |
| <u><code></u> | Defines a piece of computer code |
| <u></u> | Defines text that has been deleted from a document |
| <u><dfn></u> | Represents the defining instance of a term |
| <u></u> | Defines emphasized text |
| <u></u> | Not supported in HTML5. Use CSS instead. |

H1

3. Etiquetas HTML

3.3. Estructura del documento

- Si queremos usar caracteres especiales o reservados debemos usar:

&referencia;

Existen diccionarios de referencias UNICODE como en la web del [enlace](#)

```
Corazón    &#10084;  
Smily face &#128517;  
Flecha     &#10145;
```

Salida por
navegador

```
Corazón ♥  
Smily face 😊  
Flecha ➡
```


3. Etiquetas HTML

3.3. Estructura del documento

Tablas en HTML

En cualquier documento HTML podemos crear tablas para mostrar información como un listado de productos, un horario, un listado de tareas, etc.

Para ello, haremos uso de la etiqueta `<table></table>` que nos indica que estamos construyendo una tabla y de las etiquetas `<td></td>` y `<tr></tr>` que hacen referencia a las columnas y a las filas respectivamente.

```
<table>
  <tr>
    <td>Nombre</td>
    <td>Apellidos</td>
  </tr>
  <tr>
    <td>Samuel</td>
    <td>Moreno</td>
  </tr>
</table>
```

Salida por
navegador

| | |
|--------|-----------|
| Nombre | Apellidos |
| Samuel | Moreno |

3. Etiquetas HTML

3.3. Estructura del documento

Tablas en HTML

Del mismo modo que en el resto del documento usamos las etiquetas <h...> para definir títulos, se recomienda el uso de la etiqueta <th></th> para definir cuál será nuestra fila de cabeceras. Para mostrar las líneas hemos incluido el atributo border, aunque no se debe usar al no ser HTML5. Ya le daremos formato con CSS.

```
<table border="1">
  <tr>
    <th>Nombre</th>
    <th>Apellidos</th>
  </tr>
  <tr>
    <td>Samuel</td>
    <td>Moreno</td>
  </tr>
</table>
```

Salida por
navegador

| Nombre | Apellidos |
|--------|-----------|
| Samuel | Moreno |

3. Etiquetas HTML

3.3. Estructura del documento

Tablas en HTML

En ocasiones, nos resultará útil combinar celdas o filas que contengan un mismo valor. Para ello, podemos usar los atributos de la etiqueta `<td>` **colspan** y **rowspan**.

```
<table border="1">
  <tr>
    <th colspan="3">Profesores</th>
  </tr>
  <tr>
    <th>Nombre</th>
    <th>Apellidos</th>
    <th>Departamento</th>
  </tr>
  <tr>
    <td>Samuel</td>
    <td>Moreno</td>
    <td rowspan="2">Informática</td>
  </tr>
  <tr>
    <td>Mariano</td>
    <td>Utrillas</td>
  </tr>
</table>
```

Salida por
navegador

| Profesores | | |
|------------|-----------|--------------|
| Nombre | Apellidos | Departamento |
| Samuel | Moreno | Informática |
| Mariano | Utrillas | |

3. Etiquetas HTML

3.3. Estructura del documento

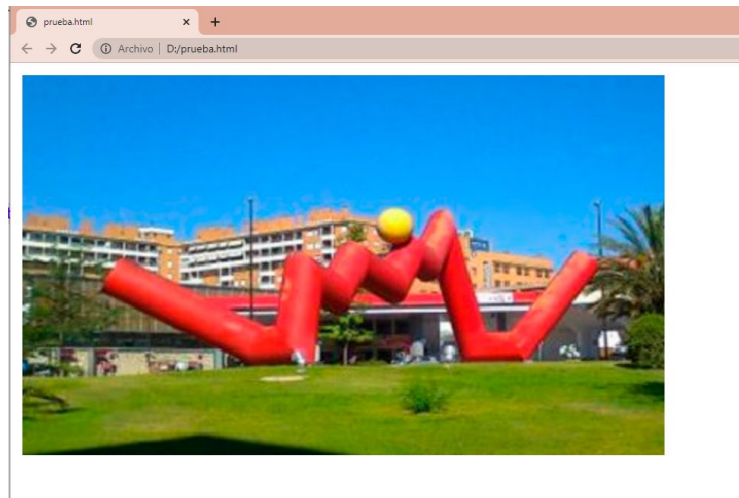
Imágenes en HTML

- Para incrustar imágenes en nuestro documento web debemos usar la etiqueta ``. El atributo `alt` se mostrará si la imagen no se muestra o para indicar a buscadores el tema de la imagen.

```

```

Salida por
navegador



3. Etiquetas HTML

3.3. Estructura del documento

Imágenes en HTML

- **Resolución de la imagen:** NO tiene sentido utilizar imágenes de más de 2 Mpx (1000 x 2000) ya que ralentizará la carga de la página además de que no mejorará la calidad de visualización de la misma.
- Para indicar el **tamaño de las imágenes en la pantalla** lo indicaremos con atributos CSS. tamaño en horizontal (ancho) width y en vertical (alto) con height.
 - En píxeles: se indica el nº y se pueden añadir las unidades (px) o no
 - En porcentaje: después del porcentaje añadir el carácter %
- **Fuente de las imágenes:**
 - Las podemos tener alojadas en una carpeta junto a la web (archivo html)
 - Podemos usar la URL de localización de la misma. En búsquedas de Google botón derecho ratón seleccionando “Copiar dirección de imagen”.

3. Etiquetas HTML

3.3. Estructura del documento

Multimedia:

Podemos insertar también un audio (como música de fondo de nuestra web) o un vídeo para que se muestre directamente en nuestra web.

Vídeo: podemos usar dos sintaxis, la que nos ofrece Youtube por ejemplo:



```
<iframe width="560" height="315"
src="https://www.youtube.com/embed/GQbyt
Kkt0tg" frameborder="0"
allow="accelerometer; autoplay;
encrypted-media; gyroscope; picture-in-picture"
allowfullscreen>
```



3. Etiquetas HTML

3.3. Estructura del documento

Multimedia:

Vídeo: o usar la sintaxis para insertar un archivo de vídeo

```
<video width="400" controls>  
  <source src="DOVER - serenade.mp4"  
    type="video/mp4">  
</video>
```



3. Etiquetas HTML

3.3. Estructura del documento

Multimedia:

Audio: Podemos insertar un reproductor de un determinado audio

```
<audio controls>
  <source src="DOVER - Serenade.mp3"
    type="audio/mpeg">
</audio>
```



O hacer que este audio funcione como música de fondo (no se mostraría nada por pantalla)
En el código ponerlo al final del todo.

- Para chrome y Edge (*):

```
<embed src="DOVER - Serenade.mp3" autostart="true" loop="true" width="0" height="0"/>
```

- Para Firefox y Explorer (desbloqueando el audio):

```
<audio src="temple of love.mp3" autoplay="autoplay" loop="loop"/>
```


3. Etiquetas HTML

3.3. Estructura del documento

Enlaces en HTML

- Una de las bases de la web son los enlaces o hipervínculos que permiten la navegación entre documentos, entre secciones de un mismo documento o direccionar a sitios web.
- Usaremos la etiqueta `<a>` junto a su atributo "href" para definir enlaces sobre elementos de nuestro documento tales como texto, imágenes o botones (button)

```
<a href="https://cpilosenlaces.com/">CPIFP Los Enlaces</a>           <br/><br/>  
<a href="https://cpilosenlaces.com/"></a> <br/><br/>  
<a href="https://cpilosenlaces.com/"><button>CPIFP Los Enlaces</button></a>
```



[CPIFP Los Enlaces](https://cpilosenlaces.com/)



[CPIFP Los Enlaces](https://cpilosenlaces.com/)

3. Etiquetas HTML

3.3. Estructura del documento

Enlaces en HTML

- En el ejemplo anterior hemos visto cómo navegar hacia otro documento html a través de URL, es posible además navegar hacia otros puntos de nuestro documento a través de las llamadas anclas.
 - Para ello crearemos un ancla en cualquier parte de nuestro documento

```
<a id="ancla"></a>
```

- Y posteriormente definiremos un enlace hacia la misma

```
<a href="#ancla">Enlace al ancla</a>
```

3. Etiquetas HTML

3.3. Estructura del documento

Enlaces en HTML

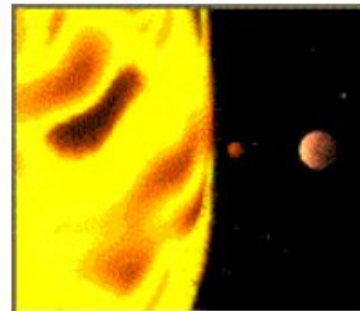
- Uso de una imagen con zonas clicables como enlaces. La sintaxis es como la del ejemplo:

```


<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" alt="Sun"
    href="sun.htm">
  <area shape="circle" coords="90,58,3" alt="Mercury"
    href="mercur.htm">
  <area shape="circle" coords="124,58,8" alt="Venus"
    href="venus.htm">
</map>
```



Click on the sun or on one

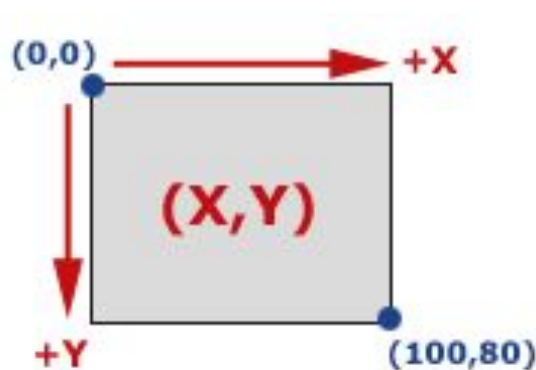


3. Etiquetas HTML

3.3. Estructura del documento

Enlaces en HTML

Las áreas pueden ser:



| FORMA | COORDENADAS |
|-------------|---|
| Rectangular | “ rect ” Posiciona el cuadrado respecto al eje x, luego al eje y, posteriormente indica hasta qué pixel llega en x e y |
| Circular | “ circle ” Posiciona el centro del círculo en el eje x e y respectivamente y posteriormente indica el radio |
| Poliédrica | “ poly ” o “ polygon ” Posiciona distintas coordenadas del perímetro de la zona cliqueable |

Consejo: para averiguar las coordenadas puedes usar aplicaciones como Paint o Gimp

3. Etiquetas HTML

3.3. Estructura del documento

Listas en HTML

- Uno de los modos más habituales para presentar una enumeración de elementos es la creación de listas. En HTML encontramos dos tipos de listas
 - **Listas desordenadas** → Las listas no ordenadas se utilizan para enumerar elementos sin orden establecido. Los elementos del listado aparecerán con una viñeta a la izquierda.

```
<ul>  
  <li>Dover</li>  
  <li>Nirvana</li>  
  <li>Roling Stones</li>  
</ul>
```



- Dover
- Nirvana
- Roling Stones

3. Etiquetas HTML

3.3. Estructura del documento

Listas en HTML

- **Listas ordenadas** → Similares a las anteriores, sin embargo, en este caso podemos establecer un orden. Con el atributo start podríamos empezar en otro nº.

```
<ol>  
  <li>Dover</li>  
  <li>Nirvana</li>  
  <li>Roling Stones</li>  
</ol>
```



```
1. Dover  
2. Nirvana  
3. Roling Stones
```

```
<ol start="0">  
  <li>Dover</li>  
  <li>Nirvana</li>  
  <li>Roling Stones</li>  
</ol>
```



```
0. Dover  
1. Nirvana  
2. Roling Stones
```

3. Etiquetas HTML

3.3. Estructura del documento

Listas en HTML

- **Listas anidadas** → HTML permite combinar y anidar varias listas, independientemente de su tipo.

```
<ul>
  <li>Dover</li>
  <ol>
    <li>Serenade</li>
    <li>Devil came to me</li>
    <li>Judas</li>
  </ol>
  <li>Nirvana</li>
  <ol>
    <li>Smell like a teen spirit</li>
    <li>Come as you are</li>
  </ol>
  <li>Roling Stones</li>
  <ol>
    <li>Satisfaction</li>
    <li>Sympathy for the Devil</li>
  </ol>
</ul>
```



- Dover
 1. Serenade
 2. Devil came to me
 3. Judas
- Nirvana
 1. Smell like a teen spirit
 2. Come as you are
- Roling Stones
 1. Satisfaction
 2. Sympathy for the Devil

3. Etiquetas HTML

3.3. Estructura del documento

Listas en HTML

Listas de descripción de términos → Usadas para listar un glosario o diccionario de términos con sus respectivos significados.

```
<dl>
  <dt>Chair</dt>
  <dd>A place to sit</dd>
  <dt>Coffee</dt>
  <dd>Black hot drink</dd>
</dl>
```



| | |
|--------|-----------------|
| Chair | A place to sit |
| Coffee | Black hot drink |

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Los formularios son la herramienta que se utiliza para recoger información concreta introducida por el usuario para su posterior procesamiento.
- Dicha información es enviada posteriormente al correo del administrador o a un servidor que contenga un programa específico capaz de procesarla. Por sí sólo, HTML no puede procesar los datos, por lo que habrá que recurrir al uso de otros lenguajes como PHP o ASP.
- Para introducir un formulario en una página Web, se utilizan las etiquetas **<form>** **</form>**. Entre ellas irán los elementos que formarán parte del mismo como texto, botones, menús desplegables, etc.

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Sus principales atributos son:
 - **action** → Indica al navegador qué debe hacer con la información una vez pulsado el botón de envío. Puede tomar el valor URL si se van a enviar los datos a un servidor (lo más usual) o el valor "mailto: direccion_de_correo", en cuyo caso deberemos añadir el parámetro *ENCTYPE="text/plain"* para que lo que recibamos resulte legible.
 - **method = " POST / GET "** → Informa al navegador sobre la forma de enviar los datos para su procesamiento. Tomará el valor **post** si el envío de información se realiza de forma no transparente y el valor **get** si los datos se envían a través de la URL. Es más seguro utilizar `post` ya que evita que la información sea captada por un intruso mirando la URL.

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- La forma en la que aparecería la etiqueta <form> de manera genérica es:

```
<form action="URL" method="post" enctype="text/plain"></form>
```

| Atributo | Valores | Significado |
|----------------|-----------|---|
| action | URL | Atributo obligatorio. Indica a dónde se debe enviar la información del formulario. |
| enctype | Tipo MIME | Determina la forma en la que se debe codificar la información, ya sea como texto plano (text/plain) o como fichero (multipart/form-data) |
| method | get, post | Determina cómo se debe enviar la información. Con get se enviará a través de la URL; con post se hará en el cuerpo de la petición (invisible a los usuarios). |
| name | Texto | Define un nombre unívoco para el formulario. |

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **Botones** → Pueden ser de *envío (submit)* o de *reinicialización (reset)*. Este último permitiría borrar los datos introducidos por el usuario y devolver los campos del formulario a sus valores iniciales.

```
<form action="URL" method="post" enctype="text/plain" name="form">  
  <input type="submit" name="enviar" value="Enviar"/>  
  <input type="reset" name="borrar" value="Restablecer"/>  
</form>
```

Enviar

Restablecer

- Imagen (para envío): en vez de botón para envío podemos usar una imagen

```
<input type="image" name="enviar" src="ok.png"/>
```



3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **Campos de texto**→ La forma más primitiva y sencilla de introducir datos en un formulario es mediante campos en los que el usuario puede escribir texto.

Esto se puede realizar mediante la etiqueta `<input>`, al menos, si el texto que se debe de introducir es corto (por ejemplo, nombre de usuario). Para ello, simplemente habrá que indicar con el atributo `type`, el valor `text`:. El valor por defecto del atributo **size** es 20 caracteres (tamaño de la caja). Se le puede dar un valor por defecto con el atributo **value** y obligar a un máximo de caracteres con el atributo **maxlength**.

```
<input type="text" name="nombre"/>
```

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **TextArea**→ A diferencia de los textboxes, con un textarea permitimos al usuario escribir un texto multilínea.
 - Podemos ajustar el ancho y el alto mediante los atributos cols y rows respectivamente (también en CSS)

```
<label>Observaciones</label>  
<textarea name="observaciones" cols="20" rows="5"></textarea>
```

Observaciones



3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **Campos de texto.**
 - Cuando los cuadros de texto van a ser usados para pedir datos secretos o “sensibles” (como contraseñas), es conveniente utilizar *type=“password”*. Así el texto tecleado por el usuario será remplazado por un conjunto de asteriscos u otros símbolos.

```
<label>Usuario:</label><input type="text" name="nombre"/><br/>  
<label>Contraseña:</label><input type="password" name="contraseña"/><br/>
```

Usuario:

Contraseña:

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **Patrón** obliga que el dato insertado siga un patrón (por ejemplo para un DNI, teléfono o CC). Con el atributo **title** saldrá un mensaje de error indicando cómo debiera ser el patrón.

```
<label>DNI:</label>  
<input type="text" name="DNI" pattern="[0-9]{8}[A-Z]" title="12345678-A"/>
```

DNI:

25171435-M

```
<input type="text" name="DNI" pattern="[0-9]{8}[A-Z]" placeholder="12345678A"/>
```

DNI:

3. Etiquetas HTML

3.3. Estructura del documento

Ejemplos de [expresiones regulares](#)*:

| Elemento \d | Elemento 2 |
|-------------|-----------------------|
| a*b | b, ab, aab, aaab, ... |
| [xyz]b | xb, yb, zb |
| a?b | b, ab |
| a+b | ab, aab, aaab, ... |
| [a-c]x | ax, bx, cx |

| | |
|--------------|---------------------------------------|
| [a-c]x | ax, bx, cx |
| [^0-9]x | Carácter ≠ dígito seguido de x |
| \Dx | Carácter ≠ dígito seguido de x |
| (pa){2}rucha | paparucha |
| .abc | Cualquier carácter (1) seguido de abc |
| (a b)+x | ax, bx, aax, bbx, abx, bax, ... |
| a{1,3}x | ax, aax, aaax |
| \n | Salto de línea |
| \p{Lu} | Letra mayúscula |
| \p{Sc} | Símbolo de moneda |

* enlace a [regexr.com](https://www.regexpalace.com/)

6. Restricciones

Ejemplos de expresiones regulares*:

| Elemento \d | Elemento 2 |
|-------------|-----------------------|
| a*b | b, ab, aab, aaab, ... |
| [xyz]b | xb, yb, zb |
| a?b | b, ab |
| a+b | ab, aab, aaab, ... |
| [a-c]x | ax, bx, cx |

| | |
|--------------|---------------------------------------|
| [a-c]x | ax, bx, cx |
| [^0-9]x | Carácter ≠ dígito seguido de x |
| \Dx | Carácter ≠ dígito seguido de x |
| (pa){2}rucha | paparucha |
| .abc | Cualquier carácter (1) seguido de abc |
| (a b)+x | ax, bx, aax, bbx, abx, bax,... |
| a{1,3}x | ax, aax, aaax |
| \n | Salto de línea |
| \p{Lu} | Letra mayúscula |
| \p{Sc} | Símbolo de moneda |

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

Elementos de un formulario:

- **Cuadros de validación (checkbox)** → Pueden ser seleccionados o deseleccionados con un simple clic. Cuando se envía la información a través de action, se incluirá la variable con el nombre que se haya indicado en la etiqueta <input>.

```
<label>¿Qué deportes practicas?</label><br/>
<input type="checkbox" name="futbol" value="futbol"/><label>Fútbol</label><br/>
<input type="checkbox" name="escalada" value="escalada"/><label>Escalada</label><br/>
<input type="checkbox" name="esqui" value="esqui"/><label>Esquí</label><br/>
<input type="checkbox" name="tenis" value="tenis"/><label>Tenis</label><br/>
```

¿Qué deportes practicas?

☐ Fútbol

☐ Escalada

☐ Esquí

☐ Tenis

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **RadioButtons** → Similares a los checkboxes pero con la particularidad de que sólo podremos seleccionar una opción de cada “Grupo”.

```
<label>¿Qué deportes practicas?</label><br/>
<input type="radio" name="estudios" value="eso"/><label>E.S.O.</label><br/>
<input type="radio" name="estudios" value="bach"/><label>Bachillerato</label><br/>
<input type="radio" name="estudios" value="fp"/><label>Formación Profesional</label><br/>
<input type="radio" name="estudios" value="uni"/><label>Universidad</label><br/>
```



Ojo! En este caso el nombre de cada input debe ser el mismo siempre.

Nivel de estudios terminado:

- ☐ E.S.O.
- ☐ Bachillerato
- ☐ Formación Profesional
- ☐ Universidad

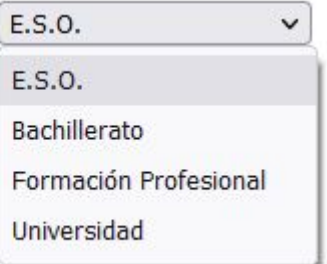
3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **Listas desplegables (ComboBox)** → Con el elemento `<select>` se definen menús desplegables con un conjunto de opciones indicadas por la etiqueta `<option></option>`.

```
<label>Nivel de estudios terminado:</label>
<select name="formacion">
  <option value="eso">E.S.O.</option>
  <option value="bach">Bachillerato</option>
  <option value="fp">Formación Profesional</option>
  <option value="uni">Universidad</option>
</select>
```

Nivel de estudios terminado: 

The dropdown menu displays the following options:

- E.S.O.
- E.S.O.
- Bachillerato
- Formación Profesional
- Universidad

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **Listas desplegables con datalist**

```
<label>Nivel de estudios terminado:</label>
<input list="formacion">
<datalist id="formacion">
  <option value="E.S.O.">
  <option value="Bachillerato">
  <option value="Formación profesional">
  <option value="Universidad">
</datalist>
```

Nivel de estudios terminado:

- E.S.O.
- Bachillerato
- Formación profesional
- Universidad

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **Color** sirve para poder elegir un color.

```
<label>Selecciona tu color favorito:</label>  
<input type="color" name="mi_color" value="#ff0000"/>
```

Selecciona tu color favorito:



3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **Fecha** sirve para poder introducir una fecha de forma sencilla.

```
<label>¿Cuál es tu fecha de nacimiento?</label>  
<input type="date" name="fecha"/>
```



diciembre de 2021 >

| lun | mar | mié | jue | vie | sáb | dom |
|-----|-----|-----|-----|-----|-----|-----|
| 29 | 30 | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **Archivo** sirve para poder seleccionar un archivo.

```
<label>Sube una foto tuya:</label>  
<input type="file" name="foto"/>
```

Sube una foto tuya: foto.PNG

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **Baremo** sirve para poder solicitar un valor cuantitativo de opinión.

```
<label>Valora la dificultad del examen:</label><br/>
<label>Fácil</label>
<input type="range" name="valoracion" min="0" max="10"/>
<label>Difícil</label>
```

Valora la dificultad del examen:

Fácil



Difícil

Valora la dificultad del examen:

Fácil



Difícil

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Elementos de un formulario:
 - **Number** nos permite obtener un valor numérico

```
<label>Edad</label>  
<input type="number" id="edad" name="edad" min="0" max="120">
```

Edad

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Atributos de la etiqueta input

| Atributo | Valores | Significado |
|-----------|----------------------------------|---|
| accept | Tipos MIME | Indica el tipo MIME del archivo (sólo se utiliza con type="file") |
| align | Left, right, top, middle, bottom | Define la alineación del texto respecto a la imagen. |
| alt | Texto | Establece un texto alternativo para la imagen. |
| checked | | Indica que el cuadro de validación debe aparecer seleccionado por defecto. |
| maxlength | Número | Determina el número máximo de caracteres que se pueden insertar en un campo de texto. |
| name | Texto | Atributo obligatorio. Define un nombre unívoco para el elemento. |
| size | Número | Define el tamaño del elemento <input>. No se puede utilizar con type="hidden" |

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Atributos de la etiqueta input

| Atributo | Valores | Significado |
|----------|---|--|
| src | URL | Indica la URL de la imagen. |
| type | Button, checkbox, file, hidden, image, password, radio, reset, submit, text | Indica el tipo de elemento. Por defecto, será texto. |
| value | | Define el valor del tipo de elemento: <ul style="list-style-type: none">• Si type="button", "reset" o "submit", define el texto del botón.• Si type="checkbox" o "radio", este atributo es obligatorio. Define el resultado del elemento input al ser pulsado. El resultado se envía a la URL indicada en <form>• Si type="hidden", "password" o "text", define el valor por defecto del elemento. |

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Otras etiquetas para formularios

| Etiqueta | Atributo | Valores | Significado |
|----------|----------|-----------------------------|---|
| textarea | cols | Número | Atributo obligatorio. Indica el número de columnas visibles. |
| | rows | Número | Atributo obligatorio. Indica el número de filas visibles. |
| | name | Nombre | Especifica el nombre del elemento. |
| | readonly | | Indica que el usuario no puede modificar el contenido del elemento. |
| button | name | Nombre | Especifica un nombre para el botón. |
| | type | Button, reset, submit | Define el tipo de botón. |
| | value | Cualquier botón | Especifica el valor inicial del botón. (Texto que aparece en el botón) |

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML

- Otras etiquetas para formularios

| Etiqueta | Atributo | Valores | Significado |
|----------|----------|---------|--|
| select | name | Nombre | Especifica un nombre para el menú desplegable. |
| | multiple | | Indica que es posible seleccionar diversas opciones a la vez. |
| | size | Número | Define el número de opciones visibles en el menú desplegable. |
| option | selected | | Indica que la opción que aparecerá seleccionada por defecto. |
| | value | Texto | Define el valor de la opción que debe ser enviada al servidor. |

3. Etiquetas HTML

3.3. Estructura del documento

Formularios en HTML (Ejemplo)

Restricciones a la entrada de datos

Ejemplo

Título sobre borde <legend>

Borde para
formulario
<fieldset>

Ejemplo de formulario: Matriculación en el ASIR

Nombre: Apellidos: Sexo: ☐ M ☐ H

Fecha de nacimiento: día mes año

Escoge los módulos en los que te matriculas:

- | | |
|---|--|
| <input type="checkbox"/> Fundamentos de Hardware | <input type="checkbox"/> Servicios de Red e Internet |
| <input type="checkbox"/> Gestión de Bases de Datos | <input type="checkbox"/> Administración de sistemas Operativos |
| <input type="checkbox"/> Implantación de Sistemas Operativos | <input type="checkbox"/> Implantación de Aplicaciones Web |
| <input type="checkbox"/> Planificación y Administración de Redes | <input type="checkbox"/> Administración de Sistemas Gestores de Bases de Datos |
| <input type="checkbox"/> Lenguajes de Marcas y Sistemas de Gestión de Información | <input type="checkbox"/> Seguridad y Alta Disponibilidad |
| <input type="checkbox"/> Formación y Orientación Laboral | <input type="checkbox"/> Empresa e Iniciativa Emprendedora |
| <input type="checkbox"/> Proyecto de Administración de Sistemas Informáticos en Red | <input type="checkbox"/> Formación en Centros de Trabajo |

Estudios previos:

Enviar

Restablecer

Pulsa el botón de "enviar" para formalizar la matrícula o el boton de "restablecer" para limpiar el formulario.

3. Etiquetas HTML

3.3. Estructura del documento

3.4 iFrames en HTML

- Iframe (inline frame o marco incorporado en inglés) es un elemento HTML que permite insertar o incrustar un documento HTML dentro de un documento HTML principal.

```
<body>
  <h2> HTML principal</h2>
  <iframe src="http://www.cpilosenlaces.com/" width=290 height=250>
    Texto para cuando el navegador no conoce la etiqueta iframe
  </iframe>
</body>
```

Gracias por participar



3. Etiquetas HTML

3.3. Estructura del documento

3.4 iFrames en HTML

- Atributos

| Atributo | Descripción |
|-------------|--|
| src | Para indicar la página web que se mostrará en el espacio del frame flotante. |
| frameborder | Para definir si queremos o no que haya un borde en el frame. Los valores posibles son 0 1. frameborder=0 indicaría que no queremos borde y frameborder=1 que sí. |
| scrolling | indica si se quiere que aparezcan barras de desplazamiento para ver los contenidos del iframe completo, en el caso que no aparezcan en el espacio reservado para el iframe. Los valores posibles son: yes no auto |
| marginwidth | para definir el margen a izquierda y derecha que debe tener la página que va dentro del iframe, con respecto al borde. Este margen va en pixels, pero prevalecerá el margen que pueda tener asignada la página web que mostremos en el frame flotante. Similar para marginheight |

3. Etiquetas HTML

3.3. Estructura del documento

3.5 Otros

- **Comentarios:** Los comentarios en un lenguaje de programación no son interpretados en la ejecución del programa, tan solo forman parte de la documentación interna del código para aclarar y facilitar la lectura del mismo.

```
<!--Comentario en HTML-->
```

3. Etiquetas HTML

Etiquetas en HTML5

[Enlace](#) al listado de etiquetas

Dudas y preguntas



Bibliografía

- W3school: <https://www.w3schools.com/>
- Línea de código: <https://lineadecodigo.com/html5/>
- Uniwebsidad: <https://uniwebsidad.com/libros/xhtml/>

UD 2.2 CSS. Ubicación y formatos

Contenido

1. Introducción
2. Aplicando estilos
 - 2.1. Estilos Inline
 - 2.2. Estilos Incrustados
 - 2.3. Hojas de estilos
3. Creando estilos
 - 3.1. Nombres
 - 3.2. Clases
 - 3.3. ID
4. Propiedades



1. Introducción

- Hasta el momento, hemos aprendido a crear un documento **HTML**, a organizar su estructura, y a determinar qué **elementos** son más apropiados para representar su **contenido**.
- Esta información nos permite definir un documento, pero no debería determinar la **forma** en que este se mostrará en pantalla. Desde la llegada de HTML esa tarea es responsabilidad de **CSS**.
- **CSS** (Cascade Style Sheet, hoja de estilos en cascada) es un lenguaje que facilita instrucciones que podemos usar para asignar estilos a los elementos HTML, como colores, tipos de letra, tamaños, etc.
- Los estilos se deben definir con CSS y luego asignar a los elementos hasta que logremos el diseño visual que queremos para nuestra página.

1. Introducción

- Los navegadores asignan estilos por defecto a algunos elementos HTML. Esta es la razón por la que de forma predeterminada algunos elementos tenían márgenes o generaban saltos de línea, pero otros eran definidos de forma parecida (tenían el mismo tipo de letra y colores, por ejemplo).
- Algunos de estos estilos son útiles, pero la mayoría deben ser reemplazados o complementados con estilos personalizados. En CSS, los estilos personalizados se declaran con propiedades.
- **Un estilo se define declarando el nombre de la propiedad y su valor separados por dos puntos.** El siguiente código declara una propiedad que cambia el tamaño de la letra a 24 píxeles.

```
font-size: 24px;
```

2. Aplicando estilos

- Las propiedades y las reglas definen los estilos que queremos asignar a uno o más elementos, pero estos estilos no se aplican hasta que los incluimos en el documento.
- Existen tres técnicas disponibles para este propósito.
 - Estilos inline
 - Estilos en la cabecera (Hoja de estilos interna)
 - Hoja de estilos externa

2. Aplicando estilos

Estilos inline

- Suponen la forma más sencilla (Y primitiva) de aplicar estilos CSS a un documento HTML.
- Utiliza un atributo global llamado **style** para insertar los estilos directamente en el elemento.
- Este atributo está disponible en cada uno de los elementos HTML y puede *recibir una propiedad o una lista de propiedades* que se aplicarán al elemento al que pertenece.
- Si queremos asignar estilos usando esta técnica, todo lo que tenemos que hacer es declarar el atributo style en el elemento que queremos modificar y asignarle las propiedades CSS.

2. Aplicando estilos

Estilos inline

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Este texto es el título del documento</title>
    <meta charset="utf-8">
  </head>
  <body>

    <p style="font-size: 20px; color: red;">Mi Texto</p>

  </body>
</html>
```



En el ejemplo podemos ver un elemento `<p>` con el atributo **style** y el valor **font-size: 20px; color: red;**.

Cuando el navegador lee este atributo, le asigna un tamaño de 20 píxeles al texto dentro del elemento `<p>`.

2. Aplicando estilos

Estilos inline

- Los estilos en línea son una manera práctica de probar estilos y ver cómo modifican los elementos, pero no se recomiendan para proyectos extensos.
- La razón es que el atributo style solo afecta al elemento en el que se ha declarado. Si queremos asignar el mismo estilo a otros elementos, tenemos que repetir el código en cada uno de ellos, lo cual incrementa innecesariamente el tamaño del documento, complicando su actualización y mantenimiento.
- Por ejemplo, si más adelante decidimos que en lugar de 20 píxeles el tamaño del texto en cada elemento `<p>` deber ser de 24 píxeles, tendríamos que cambiar cada estilo en cada uno de los elementos `<p>` del documento completo y en todos los documentos de nuestro sitio web.

2. Aplicando estilos

Estilos incrustados

- Una alternativa es la de insertar las reglas CSS en la **cabecera** del documento usando selectores que determinan los elementos que se verán afectados. Para este propósito, HTML incluye el elemento **<style>**.
- Dentro del elemento **style** definiremos las propiedades que deseemos para cada uno de los elementos HTML como se muestra a continuación.

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

2. Aplicando estilos

Estilos incrustados

La propiedad declarada entre las etiquetas **<style>** cumple la misma función que la declarada dentro de la etiqueta **<p>** del ejemplo anterior pero en este ejemplo no tenemos que escribir el estilo dentro del elemento **<p>** que queremos modificar porque, debido al selector utilizado, todos los elementos **<p>** del documento se ven afectados por esta regla.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Este texto es el título del documento</title>
    <meta charset="utf-8">

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

  </head>
  <body>
    <p>Mi Texto</p>
  </body>
</html>
```


2. Aplicando estilos

Estilos incrustados

- Declarar estilos en la cabecera del documento ahorra espacio y hace que el código sea más coherente y fácil de mantener, *pero requiere que copiemos las mismas reglas en cada uno de los documentos de nuestro sitio web.*

2. Aplicando estilos

Hojas de estilos

- Ubicamos el código CSS en un documento externo.
- El elemento **<link>** se usa para incorporar recursos externos al documento. Dependiendo del tipo de recurso que queremos cargar, tenemos que declarar diferentes atributos y valores.
- Para cargar hojas de estilo CSS, solo necesitamos los atributos **rel** y **href**. El atributo **rel** significa relación y especifica la relación entre el documento y el archivo que estamos incorporando, por lo que debemos declararlo con el valor **stylesheet** mientras que en **href** debemos indicar la ruta hacia nuestra hoja de estilos.

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

2. Aplicando estilos

Hojas de estilos

- Para definir una hoja de estilos, simplemente debemos crear un elemento de texto plano que incluya los estilos del mismo modo que los creábamos en el elemento <style> de nuestro HTML.

```
body {  
    background-color: powderblue;  
}  
h1 {  
    color: blue;  
}  
p {  
    color: red;
```

2. Aplicando estilos

Hojas de estilos

- **En cascada**
 - Una característica importante del CSS es que los estilos se asignan en cascada (de ahí el nombre hojas de estilo en cascada o Cascading Style Sheets en inglés).
 - Los elementos en los niveles bajos de la jerarquía heredan los estilos asignados a los elementos en los niveles más altos.
 - De esta forma, los estilos declarados en los elementos más concretos, prevalecen frente a los definidos en los más generales.
 - E.G. El tipo de letra que se mostraría en un `<p>` prevalece al definido en `<body>`.

2. Aplicando estilos

Hojas de estilos

- **En cascada**
 - Los estilos heredados de elementos en niveles superiores se pueden reemplazar por nuevos estilos definidos para los elementos en niveles inferiores de la jerarquía. Por ejemplo, podemos declarar una regla adicional para los elementos <p> que sobrescriba la propiedad font-size definida para el elemento <body> con un valor diferente.

```
body {  
    font-size: 20px;  
}  
  
p {  
    font-size: 36px;  
}
```

3. Creando estilos

Nombres (Del elemento)

- Una regla declarada con el nombre del elemento como selector afecta a todos los elementos de ese tipo encontrados en el documento.
- En ejemplos anteriores usamos el nombre p para modificar elementos <p>, pero podemos cambiar este nombre para trabajar con cualquier elemento en el documento que deseemos.
- Si en su lugar declaramos el nombre h1, por ejemplo, se modificarán todos los textos dentro de elementos <h1>.

```
h1 {  
    font-size: 20px;  
}
```

3. Creando estilos

Nombres (Del elemento)

- Si queremos **asignar los mismos estilos a elementos con nombres diferentes**, podemos declarar los nombres separados por una coma. En el siguiente ejemplo, la regla afecta a todos los elementos `<p>` y `` encontrados en el documento.

```
h1, p {  
    font-size: 20px;  
}
```

3. Creando estilos

Nombres (Del elemento)

- También podemos referenciar solo **elementos que se encuentran dentro de un elemento en particular** listando los selectores separados por un espacio. Estos tipos de selectores se llaman selectores de descendencia porque afectan a elementos dentro de otros elementos, sin importar el lugar que ocupan en la jerarquía.

```
main p {  
  font-size: 20px;  
}
```

La regla afecta solo a los elementos `<p>` que se encuentran dentro de un elemento `<main>`, ya sea como contenido directo o insertados en otros elementos.

3. Creando estilos

Nombres (Del elemento)

- Para definir atributos sobre un enlace debemos seleccionar:
 - a:link (para enlaces sin visitar)
 - a:visited
 - a:hover (cuando se pase por encima con el ratón)
 - a:active

3. Creando estilos

- A menudo es conveniente declarar todos los estilos en un archivo externo y luego cargar ese archivo desde cada documento que lo necesite, pero nos obliga a implementar diferentes mecanismos para establecer la relación entre las reglas CSS y los elementos dentro del documento que se verán afectados por las mismas.
- Existen varios **métodos para seleccionar los elementos que serán afectados por una regla CSS**. En ejemplos anteriores hemos utilizado el **nombre** del elemento, pero también podemos usar los valores de los atributos **id** y **class** para referenciar un solo elemento o un grupo de elementos, e incluso combinarlos para construir selectores más específicos.

3. Creando estilos

Atributo Id

- Las reglas anteriores afectan a los elementos del tipo indicado por el selector. Para seleccionar un elemento HTML sin considerar su tipo, podemos usar el atributo id. Este atributo es un nombre, un identificador exclusivo del elemento y, por lo tanto, lo podemos usar para encontrar un elemento en particular dentro del documento. Para referenciar un elemento usando su atributo id, el selector debe incluir el valor del atributo precedido por el carácter numeral (#).

```
#mitexto {  
    font-size: 20px;  
}
```

3. Creando estilos

Atributo Id

- De este modo, podemos aplicar este estilo a cualquier elemento de nuestro documento simplemente indicando el nombre en su Id.

```
<body>
  <h1>Welcome</h1>
  <p id="mitexto">
    Este párrafo usa el estilo miTexto.
  </p>
```

3. Creando estilos

Atributo Id

- La ventaja de este procedimiento es que cada vez que creamos una referencia usando el identificador mitexto en nuestro archivo CSS, solo se modifica el elemento con esa identificación, pero el resto de los elementos no se ven afectados.
- Sin embargo, esta es una forma muy específica de referenciar a un elemento y **se usa comúnmente con elementos estructurales, como <section> o <div>**. Debido a su especificidad, el atributo id también se usa frecuentemente para referenciar elementos desde JavaScript.
- Por tanto, en teoría un mismo Id no debe repetirse dentro de un documento HTML.

3. Creando estilos

Atributo Class

- En lugar de usar el atributo id para asignar estilos, en la mayoría de las ocasiones es mejor hacerlo con el atributo class. Este atributo es más flexible y se puede asignar a varios elementos dentro del mismo documento.
- Para referenciar un elemento usando su atributo class, el selector debe incluir el valor del atributo precedido por un punto.

```
.mitexto {  
    font-size: 20px;  
}
```

3. Creando estilos

Atributo Class

- Los dos primeros elementos `<p>` incluyen el atributo class con el valor "mitexto". Debido a que se puede aplicar la misma regla a diferentes elementos del documento, estos dos primeros elementos son afectados por la regla.
- Por otro lado, los dos últimos elementos `<p>` no incluyen el atributo class y, por lo tanto, se mostrarán con los estilos por defecto.

```
<body>
  <h1>Welcome</h1>
  <p class="mitexto">Frase 1</p>
  <p class="mitexto">Frase 2</p>
  <p>Frase 3</p>
  <p>Frase 4</p>
```

3. Creando estilos

Atributo Class

- Las reglas asignadas a través del atributo class se denominan clases.
- A un mismo elemento se le pueden asignar varias clases. Todo lo que tenemos que hacer es declarar los nombres de las clases separados por un espacio (por ejemplo, **class="texto1 texto2"**).
- Las clases también se pueden **declarar como exclusivas para un tipo específico de elementos declarando el nombre del elemento antes del punto**.

```
p.mitexto {  
    font-size: 20px;  
}
```


3. Creando estilos

Otro tipo de llamadas a elementos:

| | |
|----------------|--|
| * | Para seleccionar todos los elementos en general |
| A * | Para seleccionar todos los elementos que estén dentro de otro elemento o contenedor |
| A + B | Selecciona los elementos B que estén justo a continuación de un elemento A (solo 1 elemento) |
| A ~ B | Lo mismo que el caso anterior pero selecciona todos los elementos “hijo” que vayan detrás |
| A > B | Selecciona a los “hijos” B que estén dentro del elemento A |
| A:first-child | Selecciona únicamente al primer elemento del tipo A |
| A:last-child | Similar al anterior pero seleccionando al último |
| A:nth-child(n) | Seleccciona al n-ésimo elemento A |

4. Propiedades

- Las propiedades son la pieza central de CSS. Todos los estilos que podemos aplicar a un elemento se definen por medio de propiedades. Ya hemos introducido algunas en los ejemplos anteriores, pero **hay cientos de propiedades disponibles**.
- Para simplificar su estudio, se pueden clasificar en dos tipos:
 - Propiedades **de formato** → Las propiedades de formato se encargan de dar forma a los elementos y su contenido,
 - Propiedades **de diseño** → están enfocadas a determinar el tamaño y la posición de los elementos en la pantalla.
- Así mismo, las propiedades de formato se pueden clasificar según el tipo de modificación que producen. Por ejemplo, algunas propiedades cambian el tipo de letra que se usa para mostrar el texto, otras generan un borde alrededor del elemento, asignan un color de fondo, etc. En este capítulo vamos a introducir las propiedades de formato siguiendo esta clasificación.

4. Propiedades

- Para **textos**:
 - color
 - text-align: (valores: left, right, center, justify)
 - background-color
 - letter spacing
 - word-spacing
 - line-height
 - text-shadow: 3px 2px red
 - font-family (valores: serif, cursive, fantasy, monospace,...)
 - font-size (valores: small, medium, large, x-large, smaller,...)
 - font-style: (valores: normal, italic, oblique,...)
 - font-weight: (valores: lighter, bold, bolder, normal,...)
 - text-decoration: (valores: underline, overline, line-through,...)
 - text-transformation: (valores: capitalize, lowercase, uppercase,...)

4. Propiedades

- Para **tablas, contenedores, imágenes, botones, iframes y líneas**:
 - border: 15px solid green
 - border-style: (valores: solid, dotted, dashed, double, groove, ridge, inset, outset)
 - border-collapse: (valores: separate, collapse)
 - padding
 - border-spacing
 - width
 - height
 - border-radius
 - background-color
 - background-image: url(image.jpg)
 - opacity
 - color
 - vertical-align: (valores: baseline, text-top, text-bottom, super, sub,...)

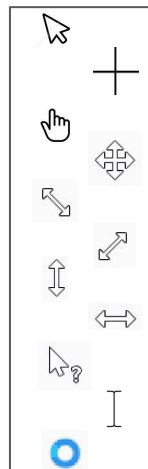
4. Propiedades

- Para **listas**:
 - list-style-type: (valores: none, circle, disc, square, upper-roman, lower-alpha)
 - list-style-image: url(image.jpg)
 - padding
 - margin-left
- Otros:
 - Cambiar cursor cuando se pase por encima de un elemento:

```

<h4 style="cursor: default">default</h4>
<h4 style="cursor: crosshair">crosshair</h4>
<h4 style="cursor: pointer">pointer</h4>
<h4 style="cursor: move">move</h4>
<h4 style="cursor: nw-resize">nw-resize</h4>
<h4 style="cursor: ne-resize">ne-resize</h4>
<h4 style="cursor: n-resize">n-resize</h4>
<h4 style="cursor: e-resize">e-resize</h4>
<h4 style="cursor: help">help</h4>
<h4 style="cursor: text">text</h4>
<h4 style="cursor: wait">wait</h4>

```



4. Propiedades

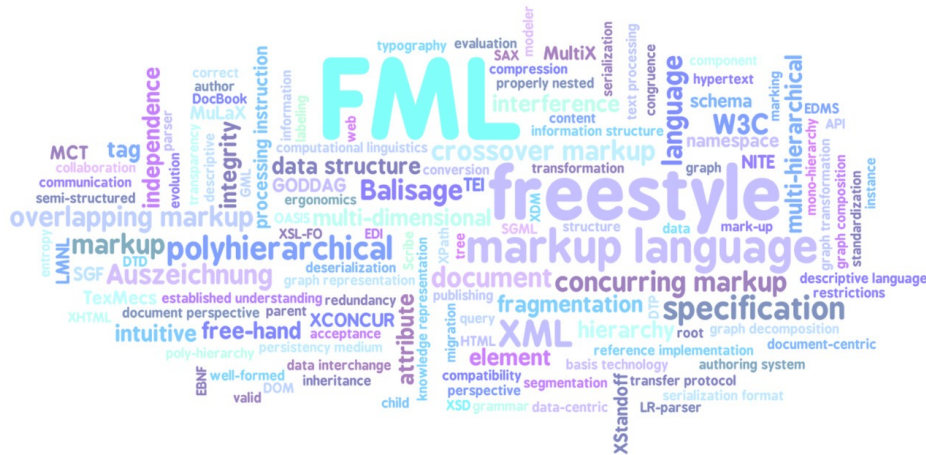
Un repositorio de propiedades podría ser el de [mclibre](#)

Y como tutorial, el de [w3school](#)

Dudas y preguntas



Lenguajes de Marcas y Sistemas de Información



UD 2.3 Maquetación web

Contenido

1. Introducción
2. Elementos estructurales
3. Articulando mi sitio web
4. Epílogo
5. Posicionamiento



1. Introducción

- Hasta el momento hemos estudiado que existen diferentes formas de maquetar los componentes de un sitio web, los primitivos **marcos (Frames) o las tablas** que hemos estado usando hasta el momento.
- En la actualidad, el uso de tablas para distribuir los contenidos de un sitio web ha caído **en desuso en pos de elementos más versátiles que nos van a permitir diseños más modernos y adaptables.**
- En este sentido el uso de **divs** (Divisiones) es la práctica más extendida a día de hoy.
 - Una `<div></div>` establece un contenedor dentro de nuestro sitio web en el que podemos incluir cualquier otro elemento (Del mismo modo que usábamos el espacio dejado por los table data)

1. Introducción

- Los navegadores crean una **caja virtual alrededor de cada elemento** para determinar el área que ocupan.
- Para organizar estas cajas en la pantalla, los elementos se clasifican en dos tipos básicos:
 - **Block (bloque)**
 - **Inline (en línea).**
- La diferencia principal entre estos dos tipos es que los elementos **Block** tienen un **tamaño personalizado y generan saltos de línea**, mientras que los elementos **Inline** **tienen un tamaño determinado por su contenido y no generan saltos de línea**. Debido a sus características,
- Los elementos Block se colocan de uno en uno en las distintas líneas, y los elementos Inline se colocan uno al lado del otro en la misma línea, a menos que no haya suficiente espacio horizontal disponible.

1. Introducción

Elementos Block

Contenido 1

Contenido 2

Contenido 3

Elementos Inline

Contenido 1

Contenido 2

Contenido 3

Contenido 4

Debido a sus características, los elementos **Block** son apropiados para crear columnas y secciones en una página web, mientras que los elementos Inline son adecuados para representar contenido.

Esta es la razón por la que los elementos que definen la estructura de un documento, como `<section>`, `<nav>`, `<header>`, `<footer>`, o `<div>`, se declaran como elementos Block por defecto, y otros como ``, ``, `<p>` o ``, que representan el contenido de esos elementos, se declaran como elementos Inline.

[Ejemplo del inside-block](#)

1. Introducción

- El que un elemento sea del tipo Block o Inline lo determina el navegador, pero podemos cambiar esta propiedad [display de CSS](#).
 - none → Elimina el elemento. (Usado por ejemplo con scripts de Java)
 - block → Muestra el elemento en una nueva línea y con un tamaño personalizado.
 - inline → Muestra el elemento en la misma línea.
 - inline-block → Es una mezcla de las propiedades de inline y block
- Los elementos estructurales se configuran por defecto con el valor block, mientras que los elementos que representan el contenido normalmente se configuran como inline.

```
header, section, main, footer, aside, nav, article, figure, figcaption{  
  display: block;  
}
```

2. Elementos estructurales

- A partir de la versión 5 de HTML se nos ofrecen una serie de elementos estructurales que nos permiten maquetar nuestra web de forma sencilla.
- **En versiones anteriores usábamos frames, tables o divs** para dar estructura a nuestros sitios web.
- Una div es básicamente una división del espacio disponible de forma que pudiéramos seccionar la web en diferentes divisiones (Una para el header, otra para el footer, otra para el contenido principal...)
- Con la llegada de **HTML5 se introdujeron elementos de estructura semántica**, estos elementos son, en esencia, divs semánticos que permiten a los navegadores y motores automatizados determinar **en qué parte de nuestra página se encuentran**.

2. Elementos estructurales

Header

- El elemento `<header></header>` permite insertar una zona de visualización para la **cabecera**.
- Puede definirse una cabecera para toda la página o para una zona determinada: artículo, main, lateral...
- Debemos considerar que este elemento puede usarse desde dos puntos de vista:
 - A nivel de página → Es la típica cabecera de la página, a menudo aparece en la parte superior, con un logotipo, un eslogan, la barra de navegación, etc.
 - A nivel de contenido → Permite disponer de una zona de introducción al contenido que se va a exponer a continuación

2. Elementos estructurales

Footer

- El elemento `<footer></footer>` permite insertar una zona de visualización para el **pie de página**.
- Del mismo modo que sucede con los encabezados, podemos definir un pie de página a nivel la página completa o bien para una zona determinada.

2. Elementos estructurales

Nav

- El elemento `<nav></nav>` está pensando para la visualización de una zona de navegación con **vínculos de hipertexto**.
- Este elemento está pensado para la navegación principal del sitio web sin embargo, podemos crear tantos elementos nav como consideremos necesarios.

2. Elementos estructurales

Section

- El elemento `<section></section>` permite **agrupar elementos** que tengan una **misma temática**.
- Podemos agrupar en un mismo elemento un contenido, con su título y su pie de página.
- Permite definir diferentes secciones de un mismo HTML

2. Elementos estructurales

Article

- El elemento `<article></article>` permite **insertar un contenido autónomo dentro de un sitio web.**
- Este contenido puede volver a usarse en otro lugar del sitio web sin que por ello se anule su significado.
- Se emplea en gran medida cuando queremos crear **contenidos cronográficos como entradas en un blog y similares.**
- Cada artículo puede tener un header, varios títulos y un pie de página.

2. Elementos estructurales

Aside

- El elemento `<aside></aside>` permite **mostrar un contenido relacionado con el contenido al cual esté vinculado.**
- Puede tratarse de **barras laterales, zonas de widgets, complementos sobre artículos, etc.**

2. Elementos estructurales

Main

- El elemento `<main></main>` permite representar el **contenido principal de la página**.
- El W3C establece una serie de restricciones para el elemento `<main>`:
 - Solo debe haber un elemento `<main>` por página.
 - No debe utilizarse dentro (elemento descendente) de:
 - `<article>`
 - `<aside>`
 - `<footer>`
 - `<header>`
 - `<nav>`

2. Elementos estructurales

Div

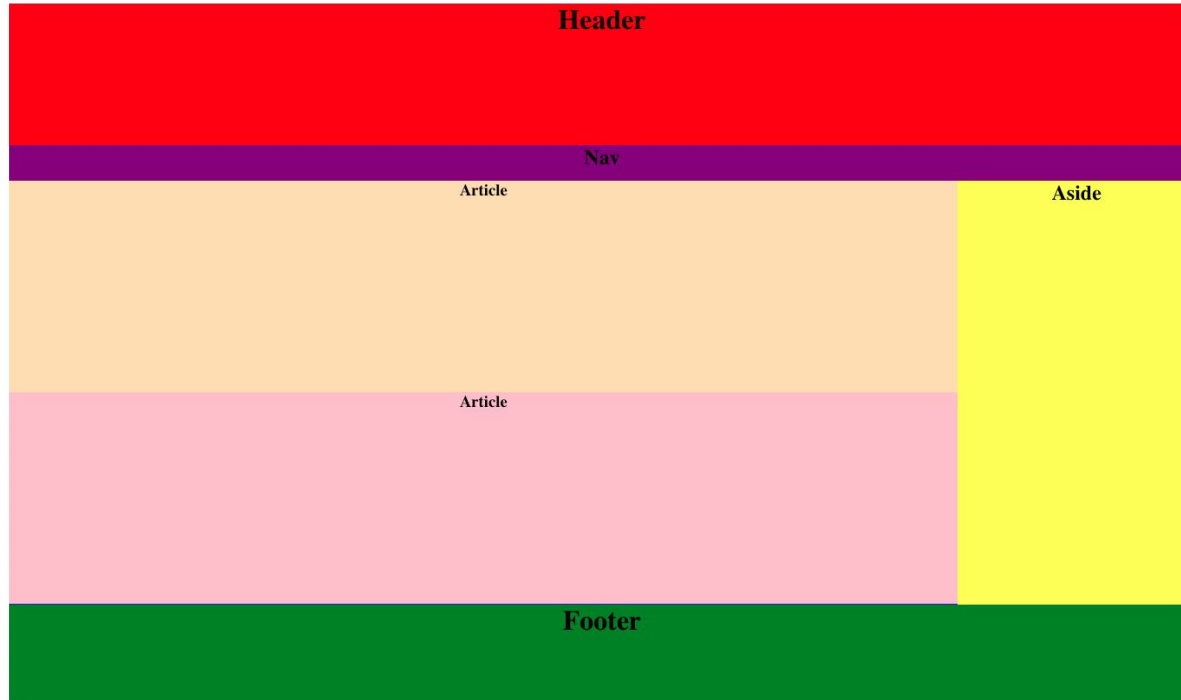
- Ya hemos mencionado el elemento `<div></div>`, aunque nos encontremos en HTML5 **no debemos erradicar su uso**, nos pueden ser de especial utilidad para casos en los que queramos establecer una caja dentro de nuestra página web pero esta no tenga porqué estar asociada a un elemento semántico:
 - Cajas flotantes.
 - Cajas de diálogo.
 - Elementos de transición.
 - Etc.

3. Articulando mi sitio web

- Una vez que hemos estudiado y conocemos los principales elementos a través de los cuales podemos maquetar una página web.
- **De forma habitual**, todas las páginas que construyamos deberían tener, al menos:
 - Un elemento `<header>`.
 - Un elemento `<nav>`
 - Un elemento `<main>`
 - Tantos elementos `<section>` y/o `<article>` como sean necesarios.
 - Un elemento `<footer>`
 - Opcionalmente podemos usar elementos `<aside>`

3. Articulando mi sitio web

- Un ejemplo básico podría ser el siguiente:



3. Articulando mi sitio web

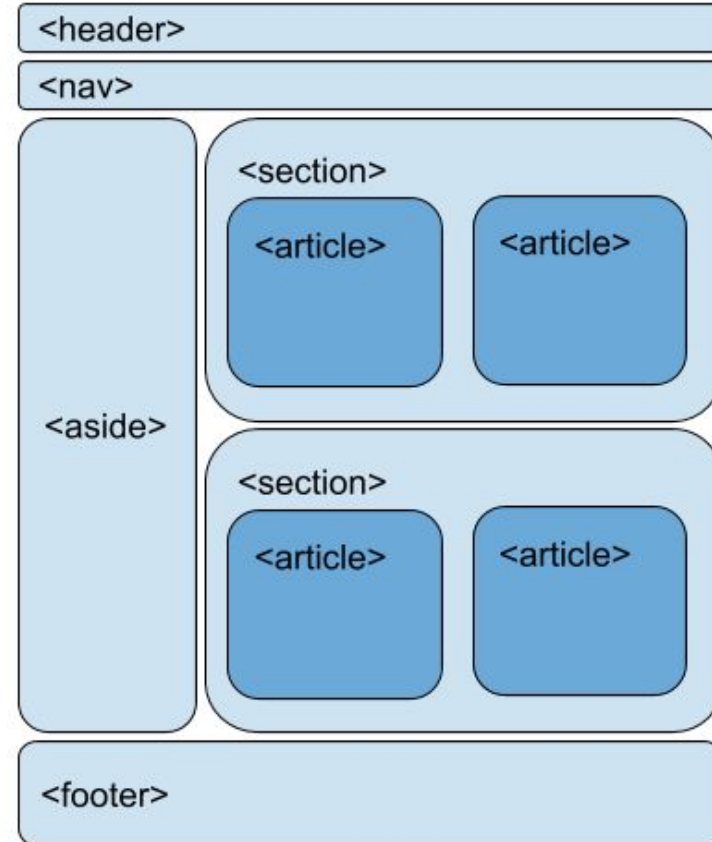
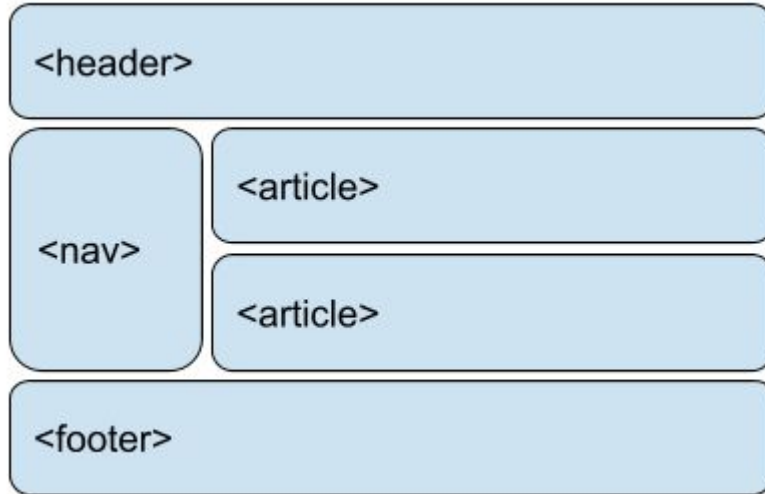
```
<body>
  <header>
    <h1> Header</h1>
  </header>
  <nav>
    <h1>Nav</h1>
  </nav>
  <main>
    <section>
      <article id="topArticle">
        <h1>Article</h1>
      </article>
      <article id="bottomArticle">
        <h1>Article</h1>
      </article>
    </section>
    <aside>
      <h1>Aside</h1>
    </aside>
  </main>
  <footer>
    <h1>Footer</h1>
  </footer>
</body>
```

```
header{
  background-color: red;
  height: 20%;
}
body{
  margin: 0;
}
main{
  background-color: blue;
  height: 60%;
}
footer{
  background-color: green;
  height: 20%;
}
aside{
  float: right;
  width: 20%;
  height: 100%;
  background-color: yellow;
}
section{
  float: left;
  width: 80%;
}
nav{
  height: 5%;
  background-color: purple;
  margin: 0;
}
#topArticle{
  background-color: navajowhite;
  height: 50%
}
#bottomArticle{
  background-color: pink;
  height: 50%
}
```

4. Epílogo

- Hemos estudiado en este documento cómo podemos estructurar los elementos en base a la utilización de las nuevas cajas semánticas que nos proporciona HTML5 y hemos visto un ejemplo de las mismas.
- Sin embargo, las posibilidades son infinitas y sólo están limitadas por la creatividad del programador.
- Podemos combinar los elementos de la forma que mejor se adapte a nuestras necesidades y crear diferentes layouts dependiendo del resultado que queramos obtener.

4. Epílogo



5. Posicionamiento

Se pueden posicionar los elementos utilizando [atributos de posicionamiento](#) que localizarán nuestra “caja” en la web con atributos que lo situarán sobre los bordes laterales.

[Ejemplos en w3school](#)

Existe un complemento para los navegadores llamado **Web developer** que ayuda a ver la salida por pantalla en distintos dispositivos para comprobar si es responsive.

Ayuda:

Para posicionar sin problemas las distintas cajas se puede usar :

```
html, body
{width: 100%;
height: 100%}
```

el valor de los atributos también podría ser auto en el caso de que la página ocupe más que el alto que se ve inicialmente.

Atributos y valores CSS más utilizados:

- **Ubicación y maquetación de contenedores:**
 - **display:** flex {distribuye los contenedores para que no se solapen}
 - **float:** right {ubicación del contenedor}
 - **height:** px o %
 - **width:** px o %
 - **border:** 2px solid black {ancho, tipo y color del borde del contenedor}
 - border-radius: 20px {para redondear las esquinas}
 - margin: 30px {espacio que deja libre alrededor el contenedor}
 - **background-color:** #referenciaColor
 - overflow: auto
- **Distribución de contenidos:**
 - **justify-content:** center {alineación del contenido en la horizontal}
 - **align-content:** stretch {alineación del contenido en la vertical, también puede tomar valores space-around o space-between}
 - flex-direction: column {organización de contenidos en horizontal o vertical}

Más recursos:

- [Animaciones básicas](#)
- [Ejemplo de animación](#)
- [Animaciones avanzadas](#)



UD 2.4 Diseño web flexible (responsive)

Contenido

1. Situación actual y objetivos
2. Las búsquedas “Media queries”
3. El tamaño de las pantallas
4. Cajas Flexibles



1. Situación actual y objetivos

- Hoy en día, una empresa que desee tener un sitio web digno de ese nombre deberá proponer un contenido que se visualice correctamente en todos los medios actuales:
 - Pantallas de ordenador
 - Tablets
 - Smartphones
- El objetivo es proponer una experiencia de uso, una interfaz web y un diseño que se adapten de forma automática a las diferentes resoluciones de pantalla que podría utilizar el usuario.
- El término en inglés que hace referencia a este tipo de arquitectura web es **Responsive Design** que podríamos traducir como **Diseño adaptable, Diseño Flexible o Diseño Sensible**.

1. Situación actual y objetivos

- Para lograr los objetivos vamos a usar tres técnicas:
 - **Un grid flexible** que permita a reorganizar la estructura de las páginas, de modo que puedan adaptarse a la resolución de pantalla.
 - **Imágenes flexibles**, para que se adapte al tamaño de la pantalla de visualización
 - **Media Queries**, que nos permitan saber cuál es el tipo y las dimensiones del dispositivo visitante.

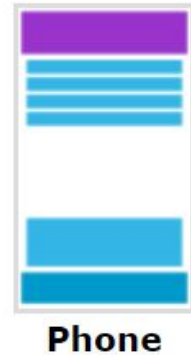
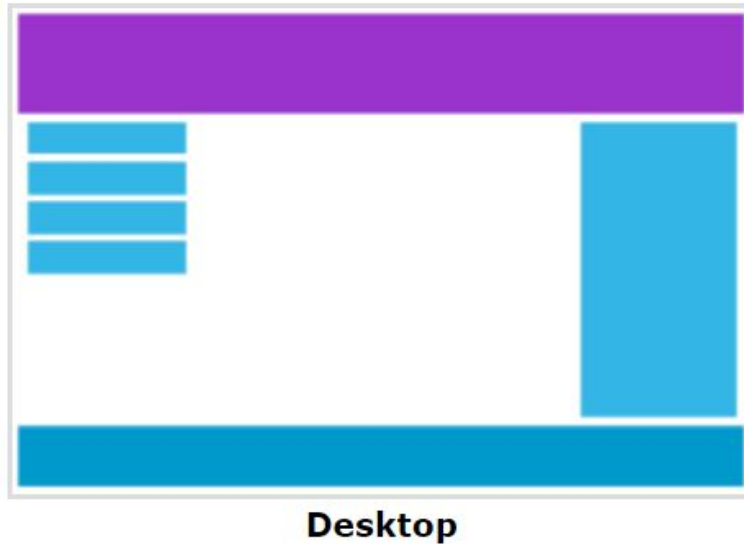
2. Las búsquedas “Media Queries”

Objetivo

- Para crear un diseño web adaptable será necesario **conocer la resolución de la pantalla utilizada**, para poder proponer estilos que se adapten a dicha resolución. Ese es precisamente el objetivo del módulo **Media Queries** que se encuentra en fase de recomendación desde el 19 de junio de 2012.
- Con las búsquedas de medios de difusión **podemos crear hojas de estilos adaptadas a los distintos tamaños de pantalla**, la elección **se hará automáticamente** y el navegador utilizará entonces el estilo adecuado

2. Las búsquedas “Media Queries”

Objetivo



2. Las búsquedas “Media Queries”

Con HTML4 y CSS 2.1

- Con HTML4 y CSS 2.1 podemos usar una hoja de estilo específica para cada medio utilizado. Simplemente tendremos que insertar el atributo elemento **media** en el elemento **link** cuando linkamos nuestra hoja de estilos.

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

```
<link rel="stylesheet" media="print" href="indexP.css">
```

2. Las búsquedas “Media Queries”

Con HTML4 y CSS 2.1

- Podemos usar:

| | |
|----------------------------------|--------------------------------------|
| screen | Para pantallas |
| handheld | Para dispositivos móviles |
| print | Para impresión |
| aural (css 2.0)/speech (css 2.1) | Para síntesis vocal |
| projection | Para uso en proyectores |
| tv | Para difusión en televisiones |
| all | Para todos los tipos de dispositivos |

2. Las búsquedas “Media Queries”

Con HTML4 y CSS 2.1

- Aunque, para la difusión en smartphones, el atributo `handheld` no es compatible con la mayoría de los navegadores, así que de momento no se puede usar.
- Esta sintaxis HTML4 / CSS 2.1 se puede usar con una regla @:

```
@media screen {  
    /*reglas css*/  
}  
  
@media print {  
    /*reglas css*/  
}
```


2. Las búsquedas “Media Queries”

Con HTML5 y CSS3

- El módulo **Media Queries** nos permite seleccionar con toda precisión el medio de difusión, mediante criterios específicos o combinaciones de criterios.
- El resultado obtenido de esta búsqueda será de tipo booleano: el valor será verdadero o falso.
- De este modo podremos seleccionar una hoja de estilo en función de las respuestas obtenidas en la búsqueda.

2. Las búsquedas “Media Queries”

Con HTML5 y CSS3

- Estos son los criterios que podremos usar en las búsquedas Media Queries:
 - El **ancho de la zona de visualización**: `width`. Podemos examinar el ancho de la zona de visualización del navegador. Ejemplo: `width: 780px`.
 - La **altura de la zona de visualización**: `height`. Podemos examinar la altura de la zona de visualización en el navegador.
 - El **ancho físico**: `device-width`. Podemos examinar el ancho físico de la pantalla de difusión.
 - La **altura física**: `device-height`. Podemos examinar la altura física de la pantalla de difusión.

2. Las búsquedas “Media Queries”

Con HTML5 y CSS3

- Estos son los criterios que podremos usar en las búsquedas Media Queries (II):
 - La **orientación de la pantalla**: `orientation`. Ejemplo: `orientation: portrait`
`orientation: landscape`. Resulta bastante práctico para examinar si el usuario usa su tableta táctil verticalmente (`portrait`) u horizontalmente (`landscape`).
 - El **ratio**: `aspect-ratio`. Para examinar el coeficiente ancho/alto. Ejemplo: `aspect-ratio: 16/9`.
 - El **ratio físico** : `device-aspect-ratio`. Para examinar el coeficiente físico ancho/alto de la pantalla.
 - El **color**: `color`. Podemos examinar si el soporte de difusión utiliza el color (valor por defecto en caso de que no se haya especificado), o el blanco y negro, o una escala de grises. Ejemplo: `min-color: 8`.

2. Las búsquedas “Media Queries”

Con HTML5 y CSS3

- Estos son los criterios que podremos usar en las búsquedas Media Queries (III):
 - El **número de colores** de la tabla de colores: `color-index`.
 - El **número de niveles de gris** para los dispositivos monocromos: `monochrome`.
 - La **resolución de la pantalla de visualización**: `resolution`. Expresada en dpi.
 - El **tipo de exploración** en las pantallas de televisión: `scan`.
 - **Grid** para ver si la pantalla de difusión utiliza una cuadrícula con un único tamaño de fuente.

2. Las búsquedas “Media Queries”

Con HTML5 y CSS3

- Sintaxis:
 - En una página HTML, en el <head>, insertaremos la búsqueda con el elemento link:

```
<link rel="stylesheet" media="screen and (width:780px)"  
href="styles780.css" />
```

- Esta sería la búsqueda en una página CSS con la regla @:

```
@media screen and (width:780px){  
    /*reglas css*/  
}
```

2. Las búsquedas “Media Queries”

Con HTML5 y CSS3

- Valores máximos y mínimos:
 - Todos los criterios que acabamos de ver, excepto orientation, scan y grid, admiten los prefijos min- y max- como valores de criterio.

```
@media screen and (max-width:780px) {  
    /*reglas css*/  
}
```

- Este criterio es muy útil, ya que no se suele trabajar para un tamaño fijo de pantalla, ya sea de ordenador, o de otro tipo.

2. Las búsquedas “Media Queries”

Con HTML5 y CSS3

- Los operadores lógicos:
 - En las búsquedas de medios de difusión podemos usar operadores lógicos para precisar y combinar diferentes criterios.

| | |
|--------|--|
| and | <code>@media screen and (min-width: 1024px) and (max-width: 1280px) { }</code> |
| or (,) | <code>@media screen and (width:780px), monochrome { }</code> |
| no | <code>@media screen and (not width:780px) { }</code> |
| only | <code>@media only screen and (width:780px) { }</code> |

2. Las búsquedas “Media Queries”

Con HTML5 y CSS3

- Búsquedas para diferentes soportes:
 - El sitio web stuffandnonsense nos propone búsquedas "listas para usar" para los diferentes tipos de pantalla:

```
/* Smartphones (portrait and landscape) ----- */  
@media only screen  
and (min-device-width : 320px)  
and (max-device-width : 480px) {  
/* Styles */  
}
```

```
/* Smartphones (landscape) ----- */  
@media only screen  
and (min-width : 321px) {  
/* Styles */  
}
```


3. El tamaño de las pantallas

- En la actualidad, cada tipo de smartphone y de tableta presenta sus propias características técnicas en cuanto al tamaño físico de la pantalla y la superficie realmente disponible para la visualización de un sitio web en la versión móvil de un navegador.
- El segundo problema está relacionado con lo que se visualizará realmente, en función de las características de la pantalla. Cuando llegó el iPhone de Apple en el 2007, el tamaño de su pantalla se estableció en 320x480 píxeles. Sin embargo, Mobile Safari muestra los sitios web con un ancho de 980 píxeles en los 320 píxeles de ancho que tiene la pantalla, lo que implica una importante disminución de la visualización del sitio web.

3. El tamaño de las pantallas

El tamaño del viewport

- Con el iPhone, Apple introdujo un nuevo atributo para el elemento meta: viewport. Este atributo nos permite controlar el coeficiente del ancho del sitio web (980 píxeles) / ancho visualizado (320 píxeles para Safari en su versión móvil).
- Si nosotros queremos que Safari para iPhone muestre un sitio web con un ancho de 320 píxeles (y no de 980 píxeles), tendremos que utilizar el atributo viewport:

```
<meta name="viewport" content="width=320" />
```

- Este atributo ha sido adoptado en la actualidad por todos los constructores de smartphones y tabletas. Se ha convertido de hecho en un estándar.

3. El tamaño de las pantallas

El tamaño del viewport

- Veamos un ejemplo simple y concreto para comprender cómo se usa viewport. Supongamos que tenemos una página simple que contiene una única caja <div> de 980 píxeles de ancho. Así se vería en un iPhone 4:

¡No se ve nada bien! La visualización de los 980 píxeles del sitio web en los 320 píxeles de la pantalla provocan una disminución importante del contenido visualizado



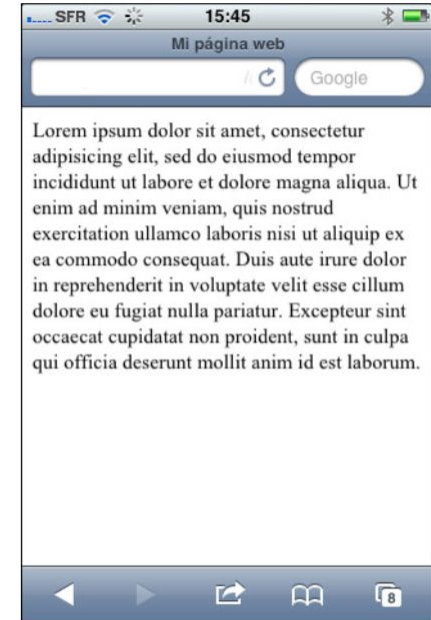
3. El tamaño de las pantallas

El tamaño del viewport

- Utilicemos ahora el atributo viewport con un valor igual al ancho de visualización de Safari para iPhone:

```
<meta name="viewport" content="width=320" />
```

Este es el resultado obtenido: ahora podemos ver la totalidad de la caja y el tamaño del texto permite la lectura.



3. El tamaño de las pantallas

El tamaño del viewport

- Todavía tenemos otro problema: el tamaño de las pantallas no está en absoluto armonizado entre los diferentes smartphones y demás tabletas táctiles. Podremos solucionarlo **indicando que se adapte la visualización en función de las características de cada dispositivo**. Para eso vamos a utilizar el valor device-width que hace referencia al ancho físico de la pantalla.

```
<meta name="viewport" content="width=device-width" />
```

- De este modo la visualización se adaptará a la superficie de visualización que proponga cada navegador específico para dispositivos móviles.

4. Cajas flexibles

Objetivos

- El módulo **CSS Flexible Box Layout Module Level 1** se encuentra en *Candidate Recommendation* dentro de los estándares de la W3C.
- Este módulo nos permite crear diseños responsive de forma sencilla, tanto para páginas web básicas como para sitios con un layout complejo.
- Los diseños flexbox han dejado obsoletos conceptos como la flotabilidad de las capas o su posición absoluta dentro de un diseño.

4. Cajas flexibles

Objetivos

- Flexbox nos permite, entre otras muchas cosas:
 - Administrar las alineaciones de las cajas
 - Centrar horizontal y verticalmente las cajas.
 - Lograr, muy fácilmente, diseños fluidos.
 - Concebir diseños responsive con un esfuerzo mínimo.

4. Cajas flexibles

Objetivos

- A pesar de ser una tecnología relativamente joven, la mayoría de navegadores ya ofrecen un soporte completo para la misma. (Can I use)

| IE | Edge | Firefox | Chrome | Safari | Opera | iOS Safari | Opera Mini | Android Browser |
|--------|-------|---------|--------|---------|---------|------------|------------|-----------------|
| | | | | | 10-11.5 | | | |
| | | 1 2-21 | 1 4-20 | 1 3.1-6 | 12.1 | 1 3.2-6.1 | | |
| 6-9 | | 3 22-27 | 21-28 | 6.1-8 | 15-16 | 7-8.4 | | 1 2.1-4.3 |
| 2 4 10 | 12-17 | 28-63 | 29-70 | 9-11.1 | 17-56 | 9-11.4 | | 4.4-4.4.4 |
| 4 11 | 18 | 64 | 71 | 12 | 57 | 12.1 | all | 67 |
| | | 65-66 | 72-74 | TP | | | | |

4. Cajas flexibles

El contenedor Flexbox

- Los diseños flexbox se albergan en contenedores flexbox. Estos contenedores le conferirá el llamado “contexto Flexbox”. Para aplicar este contexto usaremos la propiedad display de CSS.

| | |
|------------------------------------|------------------------------|
| <code>display: flex;</code> | Para las vistas en block |
| <code>display: inline-flex;</code> | Para las vistas inline-block |

4. Cajas flexibles

El contenedor Flexbox

```
<body>
  <div id="contenedor">
    <p class="bloc"> p1</p>
    <p class="bloc"> p2</p>
    <p class="bloc"> p3</p>
    <p class="bloc"> p4</p>
  </div>
</body>
```

```
#contenedor{
  display: flex;
}
p.bloc{
  background-color: #eee;
  border: solid 1px #000;
  margin-right: 10px;
}
```

p1 p2 p3 p4

4. Cajas flexibles

El contenedor Flexbox

- Como puede verse en el ejemplo anterior, en el contexto Flexbox, los elementos se muestran de forma nativa unos al lado de otros.

4. Cajas flexibles

La dirección de la vista

- La propiedad `flex-direction` permite definir la “**orientación**” de los elementos del contenedor Flexbox
 - Horizontal
 - Vertical
- Esto define el eje principal del contenedor:
 - `flex-direction: row` → Es el valor por defecto, los elementos se mostrarán unos al lado de otros horizontalmente.
 - `flex-direction: row-reverse` → Los elementos se muestran unos al lado de otros horizontalmente, pero en orden inverso a la declaración HTML.
 - `flex-direction: column` → Los elementos se mostrarán unos debajo de otros.
 - `flex-direction: column-reverse` → Los elementos se mostrarán unos debajo de otros pero en orden inverso a la declaración HTML

4. Cajas flexibles

Salto de línea o de columna

- La propiedad flex-wrap permite definir si se autoriza el **salto de línea** en los elementos hijos.
 - flex-wrap: nowrap → Es el valor por defecto. No autoriza el salto de línea
 - flex-wrap: wrap → Autoriza el salto de línea
 - flex-wrap: wrap-reverse → Autoriza el salto de línea pero en orden inverso al HTML

4. Cajas flexibles

Salto de línea o de columna

```
#contenedor1{
  display: flex;
  flex-direction: column;
  flex-wrap: nowrap;
  height: 50px;
}
#contenedor2{
  display: flex;
  flex-direction: column;
  flex-wrap: wrap;
  height: 50px;
}
p.bloc{
  background-color: #eee;
  border: solid 1px #000;
  margin: 0;
}
```

```
<body>
  <div id="contenedor1">
    <p class="bloc"> .....p1</p>
    <p class="bloc"> .....p2</p>
    <p class="bloc"> .....p3</p>
    <p class="bloc"> .....p4</p>
  </div>
  <p>&nbsp;</p>
  <div id="contenedor2">
    <p class="bloc"> .....p1</p>
    <p class="bloc"> .....p2</p>
    <p class="bloc"> .....p3</p>
    <p class="bloc"> .....p4</p>
  </div>
</body>
```

| |
|---------|
|p1 |
|p2 |
|p3 |
|p4 |

| | |
|---------|---------|
|p1 |p3 |
|p2 |p4 |

4. Cajas flexibles

La propiedad con sintaxis corta

- La propiedad flex-flow es la sintaxis corta de las propiedades flex-direction y flex-wrap.

```
flex-direction: column;  
flex-wrap: wrap;
```

```
flex-flow: column wrap;
```

4. Cajas flexibles

Alinear y centrar el eje principal

- La propiedad **justify-content** permite alinear y centrar los items en el eje principal de nuestro contenedor.
- En las siguientes definiciones, el eje principal es horizontal, es decir flex-direction: row.
 - justify-content: flex-start → Valor predeterminado. Los elementos se alinean a la izquierda del contenedor.
 - justify-content: flex-end → Los elementos se alinean a la derecha del contenedor.
 - justify-content: center → Los elementos se centran en el contenedor.
 - justify-content: space-between → Reparte uniformemente los items tomando toda la anchura del contenedor.
 - justify-content: space-around → Similar a space between pero añade un espacio antes del primer elemento y después del último.

4. Cajas flexibles

Alinear y centrar el eje secundario

- La propiedad **align-content** permite alinear y centrar los items en el eje secundario de nuestro contenedor.
- En las siguientes definiciones, el eje principal es horizontal, es decir flex-direction: row.
 - align-content: stretch → Valor por defecto. Estira verticalmente los elementos dentro del contenedor para ocupar todo el espacio.
 - align-content: flex-start → Alinea los elementos en la parte superior del contenedor.
 - align-content: flex-end → Alinea los elementos a la parte inferior del contenedor.
 - align-content: center → Centra verticalmente los elementos.
 - align-content: baseline → Alinea los elementos con la línea del texto.

4. Cajas flexibles

Propiedades para los items flexbox

- El orden de presentación → La propiedad **order** permite determinar el orden de presentación de los items, independientemente de su declaración HTML
 - Los valores de esta propiedad son números enteros.
 - Cuanto menor es el número (Se incluyen números negativos) más al inicio aparecerá nuestro elemento.
- La alineación de un ítem → La propiedad **align-self** permite establecer la alineación de cada elemento de forma independiente. Sus valores son tomados de la propiedad align-items

4. Cajas flexibles

Propiedades para los items flexbox

- Aumentar tamaño de los items → La propiedad **flex-grow** determina si el tamaño de un item puede aumentar o no dentro del contenedor. Este aumento se hace a lo alto o a lo ancho dependiendo de la dirección del eje principal.
 - flex-grow:0 → Valor por defecto, el ítem no puede aumentar su tamaño.
 - flex-grow: 2 → El item podrá ser 2 veces mayor que otro con valor 1.
- Disminuir el tamaño de un item→ La propiedad **flex-shrink** determina si se puede disminuir el tamaño de un item.
 - flex-shrink: 1 → Valor por defecto. Permite disminuir el tamaño.
 - flex-shrink: 0 → El item no puede disminuir su tamaño.

4. Cajas flexibles

Propiedades para los items flexbox

- Especificar el tamaño de un item → La propiedad **flex-basis** permite especificar el ancho o el alto de un item.
 - flex-basis: auto → Valor predeterminado. El tamaño viene dado por el contenedor padre.
 - flex-basis: valor → Toma el valor dado.
 - flex-basis: 0 → El tamaño del item no cambia.

4. Cajas flexibles

Elaboración de un diseño flexible

- [Ejemplo adjunto.](#)

| Mi sitio web | | |
|----------------------------|-------------------------------|------------------------|
| Lorem ipsum dolor sit amet | | |
| Inicio | Proyectos | Articulos |
| Conceptos | Realizaciones | Clientes |
| Ideas | Contactos | |
| Enlaces | Titulo1 | Enlaces |
| Enlace | Lorem ipsum dolor sit amet... | Enlace |
| Enlace | Titulo2 | Enlace |
| Enlace | Lorem ipsum dolor sit amet... | Enlace |
| Enlace | Titulo3 | Enlace |
| Enlace | Lorem ipsum dolor sit amet... | Enlace |
| Enlace | | |
| Qui ipsorum... | | |

