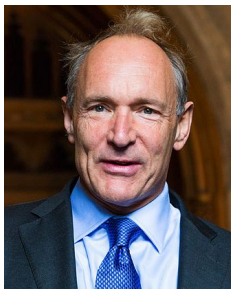
HTML

Sitio:	Aula Virtual do IES de Teis	Impreso por:	Deivid Durán Durán
Curso:	Linguaxes de Marcas e Sistemas de Xestión de Información 2021-22 (DAM-A)	Data:	Venres, 24 de Xuño de 2022, 00:15
Libro:	HTML		

Táboa de contidos

1. HTML

- **HTML:** HyperText Markup Language
- Creado por el británico Tim Berners-Lee y su grupo de trabajo en 1991 mientras trabajaba en el CERN donde creó las bases de la Wold Wide Web:
  - HTTP (*HyperText Transfer Protocol):* Protocolo de comunicación en redes cliente-servidor mediante petición-respuesta
  - URL (***U****niform* ***R****esource* ***L****ocator): Sistema de localización de recursos (páginas, ficheros, imágenes, etc.)*
  - HTML: Lenguaje de marcas para la creación de páginas web y que permite interconectar recursos mediante enlaces
- Inicialmente descrito como un subconjunto de SGML



Sir Tim arriving at the Guildhall to receive the Honorary Freedom of the City of London by Paul Clarke - CC BY-SA 4.0

2. Evolución de HTML: Versiones

HTML fue en sus orígenes un estándar a cargo del W3C, fundado también por Tim Berners-Lee en 1994

Version	**Año de publicación**
HTML+	1993
HTML2.0	1995
HTML3.2	1997
HTML4.01: Disponía de tres subestándares: Strict, Transitional, y Frameset.	1999
XHTML 1.0 (Una reformulación de HTML 4)	2000
1ª versión borrador de HTML5: nuevas etiquetas y mayor soporte para contenidos multimedia	2008

El W3C decidió estancar el desarrollo de HTML para centrarse en XHTML, más cercano a XML. Incluso llegó a trabajar en una versión XHTML 2.0 que no era compatible con XHTML 1.0.

En 2004 en el W3C se reabrió el debate de la evolución del HTML, y se dieron a conocer las bases para la versión HTML5. No obstante, este trabajo fue rechazado por los miembros del W3C y se daría preferencia al desarrollo del XML.
Sin embargo, Apple, Mozilla y Opera anunciaron su interés en seguir trabajando en HTML bajo el nombre de WHATWG que se basa en la compatibilidad con tecnologías anteriores.

En 2006, el W3C se interesó en el desarrollo de HTML5, y en 2007 se unió al grupo de trabajo del WHATWG para unificar el proyecto.

En 2011, ambas entidades llegaron a la conclusión de que tenían diferentes metas: El W3C quería publicar una versión terminada de HTML5, mientras que WHATWG quería que se mantuviese un estándar vivo y cambiante

En 2019, WHATWG y W3C firmaron un acuerdo de colaboración en una sola versión de HTML que se conoce como HTML5.

3. HTML básico

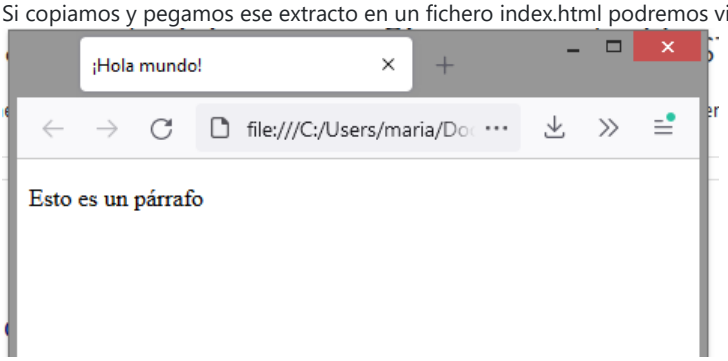
- HTML es un lenguaje de marcas con un número limitado de etiquetas.
- Las etiquetas usan < y >, al igual que XML para formar etiquetas de apertura y cierre.
- Es posible anidar elementos.
- Los documentos html pueden usar la extensión .html o .htm

Un ejemplo de un documento HTML básico sería el siguiente:

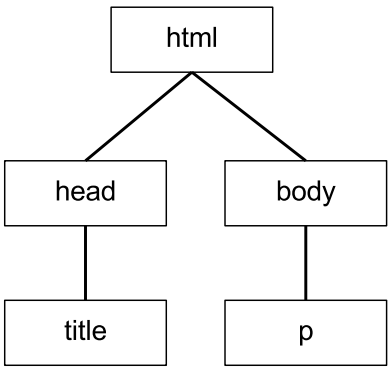
```
<!DOCTYPE html>
<html>
  <head>
     <title>¡Hola mundo!</title>
  </head>
  <body>
    <p> Esto es un párrafo </p>
  </body>
</html>
```

- `<!DOCTYPE html>` — el tipo de documento para HTML5. Aunque no es obligatorio, ayuda a identificar el tipo de documento a los navegadores.
- html es el elemento raíz.
- head encierra contenido que no será visible en el navegador, salvo el subelemento title que será el título de la ventana o pestaña del navegador. Dentro de head se pueden situar:
  - vínculos con otros ficheros como hojas de estilos, ficheros de sindicación de contenidos, etc.
  - Información sobre la codificación de caracteres, instrucciones para la visualización del navegador o las palabras claves de la página que se mostrarán en los motores de búsqueda
- body encierra el contenido de la página.
- p es una etiqueta para encerrar texto en un párrafo.

Si copiamos y pegamos ese extracto en un fichero index.html podremos visualizarlo con un navegador.



Los navegadores crean una estructura en forma de árbol llamada DOM (Document Object Model) o modelo de objetos del document con los elementos anidados del marcado.



- **Title**: A DOM tree showing the nested structure of an HTML file.

- **Creator**: Lubaochuan
- **Source:** https://commons.wikimedia.org/wiki/File:DOM_tree.svg
- **License:** "CC BY-ND 4.0"

4. HTML vs XHTML

HTML vs XHTML

La sintaxis de HTML es más flexible que la de XML y, por lo tanto, que la de XHTML.

- XHTML requiere la existencia de un DOCTYPE, ~~mientras que en HTML no es necesario, pero sí recomendable~~ y en HTML es un preámbulo requerido. A continuación se muestran diferentes DOCTYPE según la versión de HTML y XHTML:

HTML5 and beyond

```
<!DOCTYPE HTML>
```

HTML 4.01

Strict

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
```

Transitional

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
```

Frameset

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
```

XHTML 1.0

Strict (quick reference)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Algunas diferencias destacables son:

- XHTML requiere que los elementos “html”, “head”, “title” y “body” estén presentes y un único elemento raíz: html. En HTML, no siempre es necesario que exista un único elemento raíz, que se supondrá html. => Nosotros **usaremos** siempre los **elementos requeridos en XHTML** que genera el plugin Emmet de VS Code.
- En XHTML se obliga que todos los documentos deben estar correctamente anidados. En HTML esto puede no ser necesario. => Nosotros trabajaremos con **elementos correctamente anidados, como en XML**.
- En XHTML, los nombres de elementos y atributos deben escribirse en minúsculas. En HTML no es necesario => Nosotros trabajaremos siempre con **minúsculas**.
- En XHTML, los elementos deben tener etiqueta de cierre. En HTML no es necesario. Por ejemplo: <hr> o
 => Nosotros **cerraremos las etiquetas salvo para los elementos vacíos**
- En XHTML, los valores de los atributos deben ser definidos siempre entre comillas dobles o simples. En HTML, los atributos pueden no tener valor asignado y no necesitan comillas si no hay espacios en blanco en el valor del atributo. Por ejemplo: https://www.mclibre.org/consultar/htmlcss/html/html-xhtml-diferencias.html#atributos => Nosotros **usaremos siempre comillas** para encerrar el valor de los atributos.

5. Herramientas de desarrollo

HTML es un lenguaje de marcas y, por lo tanto, se escribe en texto plano y bastaría con un editor de textos sencillo con Notepad o el bloc de notas. Pero, como hemos visto, es más productivo utilizar un editor especializado. A continuación, se mencionan algunas herramientas de utilidad:

Editores

- Sublime Text
- Brackets
- IDEs como NetBeans o Eclipse.
- Nosotros seguiremos utilizando Visual Studio Code

Utilidades en VS Code

- **Plugin Emmet:** Presente en VS Code. Instalable en otros editores.

Permite utilizar abreviaturas para la creación de código HTML

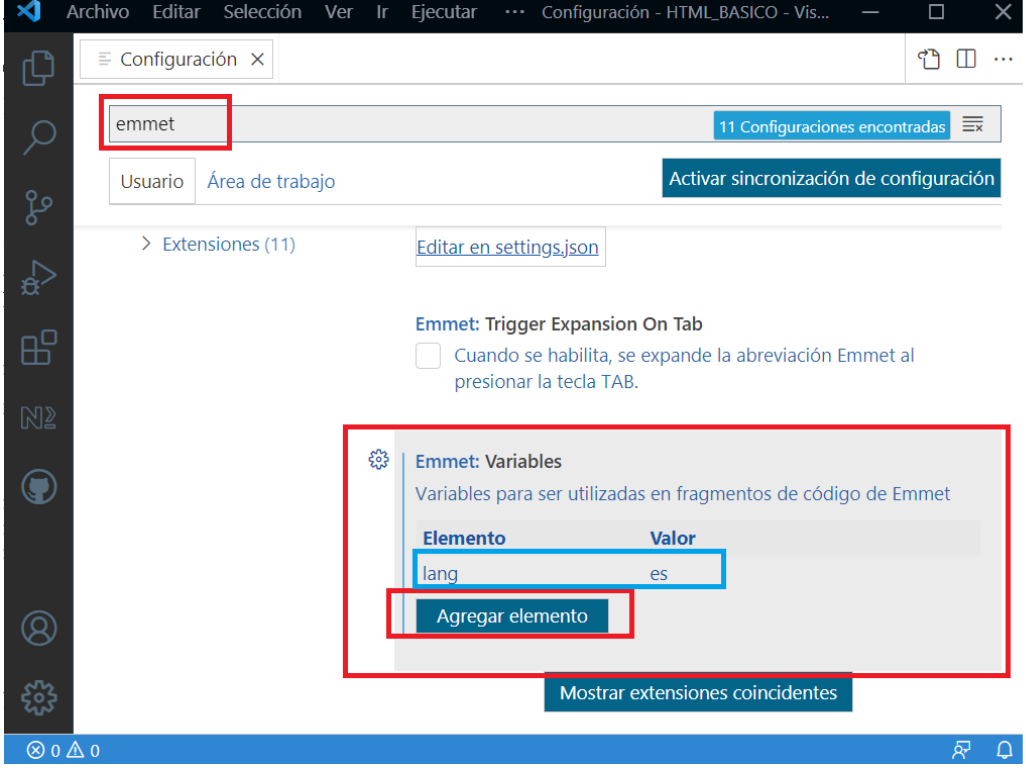
  - Si escribimos ! seguido de la tecla TAB se generará lo un documento HTML con el siguiente marcado:

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>Document</title>
8   </head>
9   <body>
10      
11  </body>
12  </html>
```

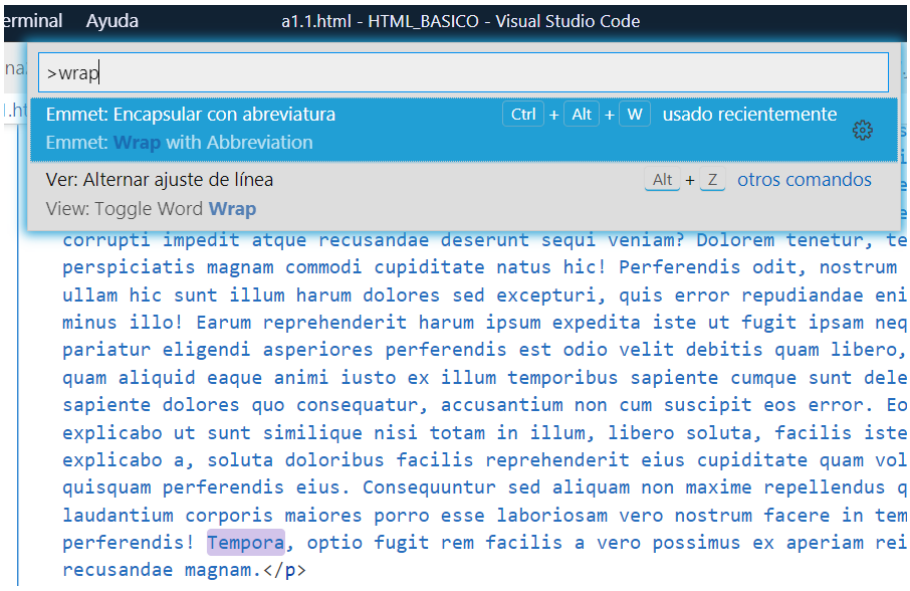
Se crean de forma automática las etiquetas recomendadas de un documento HTML básico y algunas etiquetas meta para indicar:
- línea 4: la codificación de caracteres UTF-8
- línea 5: una instrucción para que el navegador Internet Explorer use la versión más avanzada de sus modos. En la práctica equivale a usar HTML5. Se utilizaba cuando se hacían migraciones de aplicaciones para tener compatibilidad con anteriores de Internet Explorer.
- línea 6: una instrucción para el navegador para que se muestre con toda la anchura de la pantalla del dispositivo. El viewport es el área visible de la página web y varía de un dispositivo a otro. De esta forma se procura que la página sea visible con el zoom óptimo en cada dispositivo.

Más información sobre las etiquetas meta

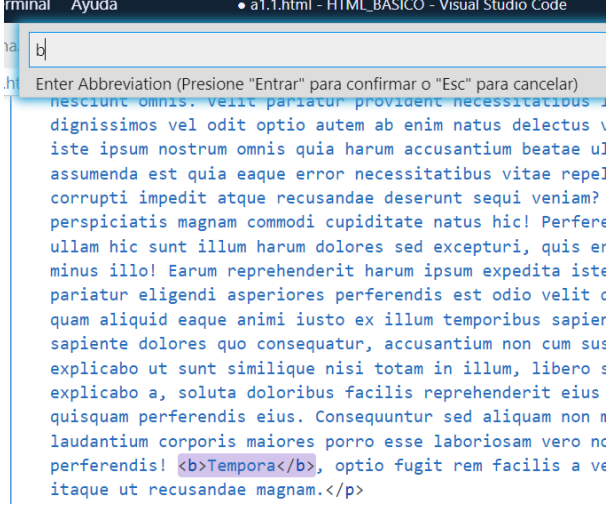
Si queremos que en lugar de inglés utilice otro idioma para las páginas Archivo > Preferencias>Configuración. Buscamos Emmet, abajo en la sección Variables, añadimos lang con valor es:



- Si escribimos p seguido de la tecla TAB o INTRO:
- Nos permite escribir texto de relleno *Lorem ipsum*
  - Si escribimos lorem, se generará un texto ficticio de 30 palabras
  - Si lo queremos incluir dentro de un párrafo: p>lorem
  - Si solo queremos 5 palabras: p>lorem5
  - Si queremos 200 palabras: p>lorem200. Para que se muestre en el área visible del navegador sin utilizar la barra de desplazamiento a la derecha: ALT+Z
  - Si queremos rodear un texto con una etiqueta: Seleccionamos el texto que deseamos rodear, botón derecho > Paleta de Comandos (o su equivalente atajo CTRL+SHIFT+P) > Escribimos wrap en la caja de texto que nos aparecerá:

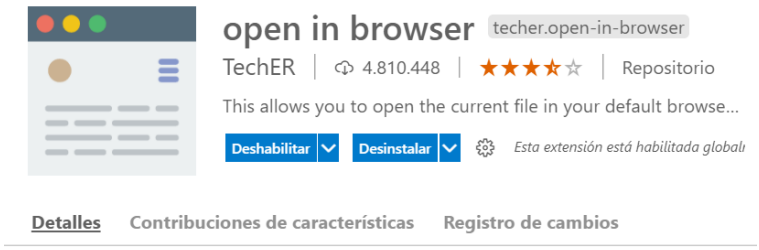


Y a continuación, la abreviatura emmet con la que deseamos rodear la selección. Por ejemplo, para rodearla con una etiqueta de bold (negrita):



También es posible asignarle un atajo a wrap mediante la rueda de engranaje, por ejemplo: CTRL+ALT+w

- Extensión para visualizar directamente en el navegador el documento HTML: Por ejemplo: **open in browser**



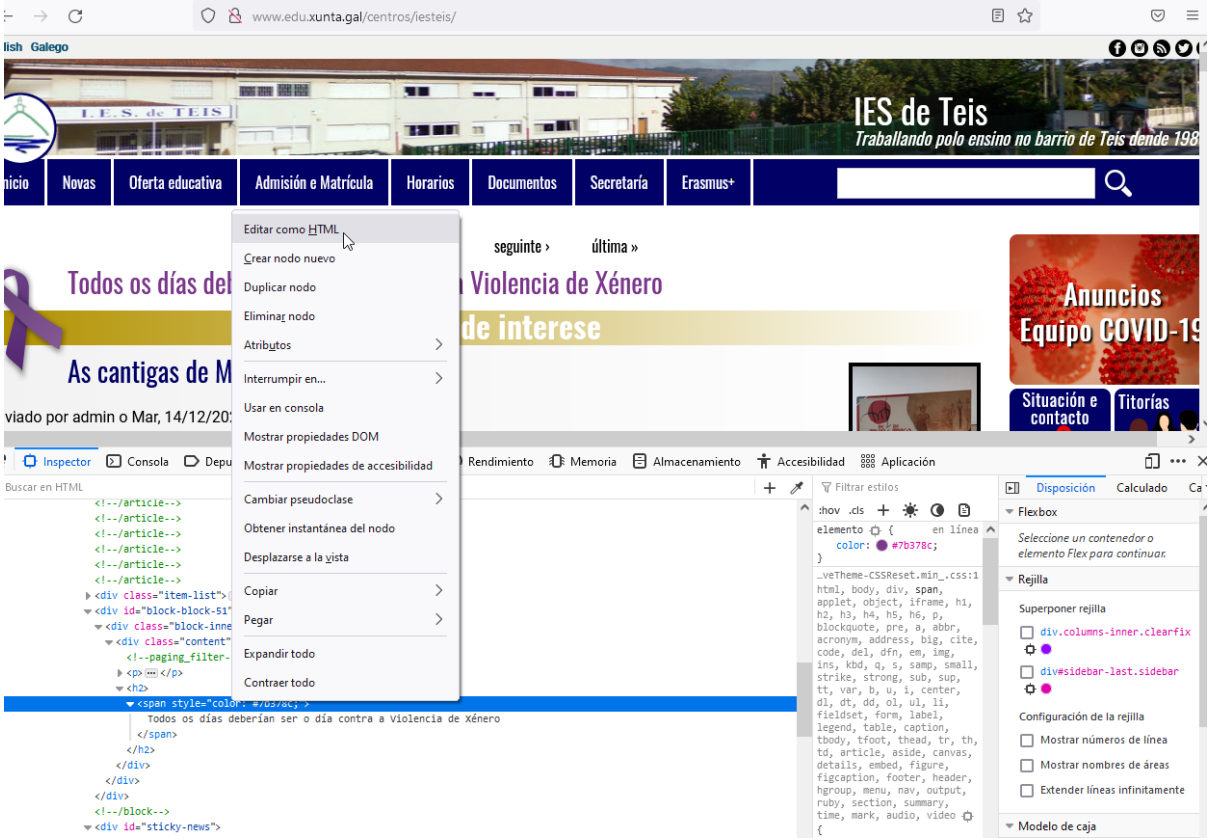
  - `Alt + B` para abrir el navegador por defecto
  - Shift + Alt + B para permitir escoger el navegador donde se abrirá el archivo actual

Herramientas del desarrollador en navegadores

Todos los navegadores cuentan con herramientas para el desarrollador que nos permiten inspeccionar el código HTML, CSS, ver información sobre las peticiones HTTP, etc.

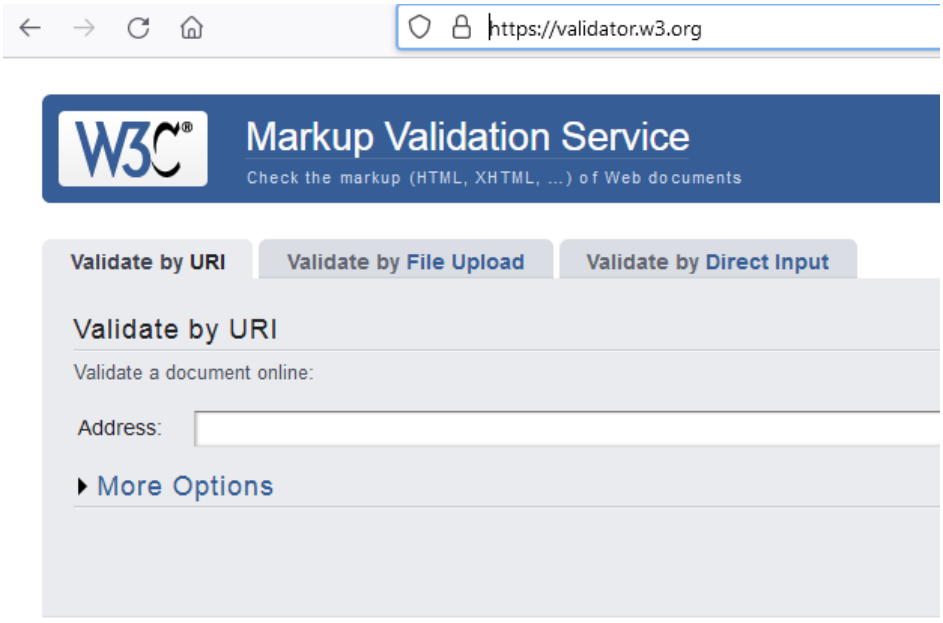
Edge, Firefox y Chrome: Botón derecho > inspeccionar o presionando la tecla F12

Nos permiten incluso editar la visualización en el propio navegador (sin afectar al fichero original).



Validador online del W3C

Para comprobar que el marcado HTML es válido podemos usar la herramienta online https://validator.w3.org/



6. Etiquetas básicas de HTML

Entre los recursos de referencia que usaremos están:

- Tutorial de HTML de w3schools
- Documentación HTML MDN de Mozilla

Encabezados

- h1, h2, ..., h6
- HTML Headings (w3schools.com)

Párrafos

- Párrafos:
  - Elementos vacíos: En HTML, al contrario de lo que ocurre con XHTML, es posible no cerrar este tipo de etiquetas.
    - <hr> horizontal rule
    -
 breaking line: Solo debe usarse para representar nuevas líneas, no para separar párrafos.
- Texto preformateado: <pre></pre> y <code></code>

Formato de texto

Como vimos, las primeras versiones de HTML estaban más orientadas a la presentación. Existen una serie de etiquetas que permiten cambiar la tipografía, pero en las versiones actuales de HTML5 se tiende a separar la presentación de la estructura del documento, por lo que, es conveniente que conozcáis las etiquetas, pero que no las uséis en HTML.
Ese tipo de cambios que afectan al formato del texto, se realizará por medio de hojas de estilos CSS.
HTML Text Formatting (w3schools.com)

Comentarios en HTML

<!--Comentarios como en XML -->

6.1. Enlaces

Recursos

- https://developer.mozilla.org/es/docs/Web/HTML/Element/a
- HTML Links Hyperlinks (w3schools.com)

Ejemplos:

```
<a href="https://www.edu.xunta.gal/centros/iesteis/">IES de Teis</a>
```

- Atributo href: Indica el destino del recurso. Puede ser una URL:
  - remota: `href="https://www.edu.xunta.gal/centros/iesteis/"`
  - local: `href="pagina2.html"`
  - absoluta: `href="https://www.edu.xunta.gal/centros/iesteis/" o href="file:///C:/Users/maria/Documents/index.html"`
  - relativa: `href="index.html"`
  - Enlaces que permiten abrir las aplicaciones de correo para enviar un correo o de llamadas para realizar llamadas a un número de teléfono:

```
<a href="mailto:nowhere@mozilla.org">Enviar correo a nowhere</a>
<a href="tel:+491570156">+49 157 0156</a>
```

- Enlaces a partes de un documento HTML: El atributo global id.

```
<a href="#p2">Ir al párrafo nº 2</a>
<p id="p2">
Lorem ipsum dolor sit amet consectetur adipisicing elit
</p>
```

- O partes de otro documento:
  `<a href="index.html#t6">Ir a index > h6</a>`

- Puede ser utilizado `href="#top"` o un fragmento vacío `href="#"` para enlazar a la parte superior de la página actual. Este comportamiento está especficado en HTML5.

Atributo target: Indica cómo se va a abrir la URL enlazada: en la misma o diferente ventana/pestaña. Principalmente veremos:
- `_self`: Carga la URL en el mismo contexto de navegación que el actual. Este es el comportamiento por defecto.
- `_blank`: Carga la URL en un nuevo contexto de navegación. Usualmente es una pestaña, sin embargo, los usuarios pueden configurar los navegadores para utilizar una ventana nueva en lugar de la pestaña.

6.2. Imágenes

Recursos:

- https://developer.mozilla.org/es/docs/Web/HTML/Element/img
- https://www.w3schools.com/html/html_images.asp

```
<img src="img_girl.jpg" alt="Girl in a jacket" width="500" height="600">
```

Atributos destacados:

- src: **Obligatorio**: URI al archivo de la imagen. Puede ser una URL externa o local. Siempre se recomienda usar rutas relativas a la ubicación de la página HTML
- alt: Texto alternativo que favorece la accesibilidad de la web.
- width y height: Un número que indica los píxeles de anchura y altura respectivamente. Si solo se indica uno de ellos, el otro conserva las proporciones. Sin ellos, la imagen se muestra en su tamaño original. Se recomienda usar un tamaño óptimo de imagen para el sitio web para reducir el tiempo de carga de imágenes.

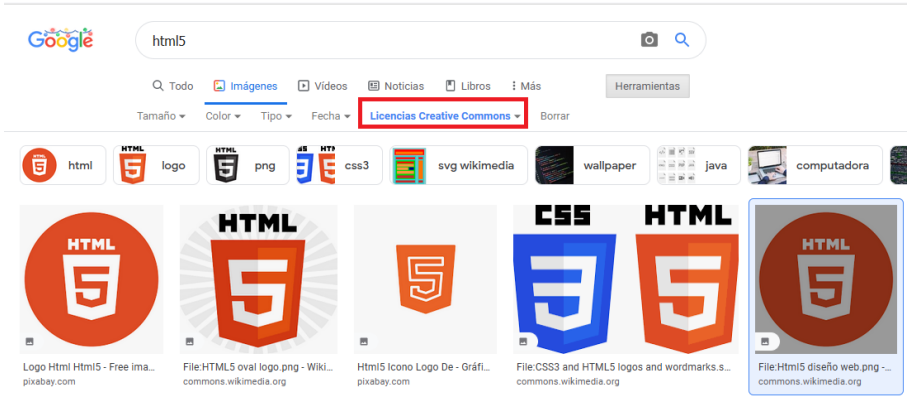
Atributos globales (comunes a cualquier elemento HTML):

- title: Se visualizará como un tooltip, un mensaje que aparecerá brevemente cuando el cursor del ratón se pose sobre el elemento.

Recursos online de imágenes

Es importante no incluir ninguna imagen que esté sujeta a derechos de autor o de la que desconozcamos su licencia. Para ello, podemos visitar algunas webs de imágenes como:

- https://pixabay.com/
- https://wordpress.org/openverse/?referrer=creativecommons.org
- https://www.pexels.com
- O buscar en Google > Imágenes > Herramientas > Licencias Creative Commons



Un ejemplo de atribución para la licencia Cretative Commons **CC BY-ND 2.0** se puede encontrar en el enlace: https://creativecommons.org/use-remix/

6.3. Listas

Listas

Recursos:

- https://www.w3schools.com/html/html_lists.asp

Existen 3 tipos diferentes de listas:

- Ordenadas (ordered lists):<ol>: https://www.w3schools.com/tags/tag_ol.asp
- No ordenadas (unordered lists): <ul>: https://www.w3schools.com/tags/tag_ul.asp

Ambos tipos de listas anidan, por cada elemento a mostrar un list item <li>. Ver ejemplo enlace

- Listas de descripciones (description lists): <dl>: En este caso anidan un término <dt> y su descripción <dd>. Ver ejemplo enlace

Tipos de listas en HTML

Listas desordenadas

- Item A
- Item B
- Item C

Listas ordenadas

1. Item 1
2. Item 2
3. Item 3

Listas de descripciones

Palabra 1
: Definición palabra 1

Palbra 2
: Definición palabra 2

Palbra 3
: Definición palabra 3

6.4. Tablas

Tablas

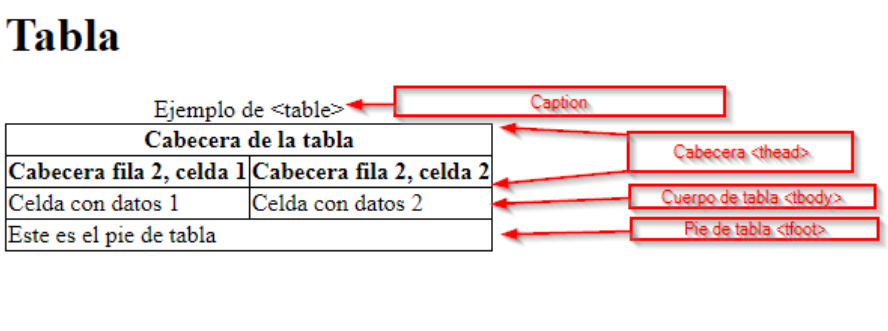
Recursos:

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/table

El elemento <table> permite mostrar información en forma tabular.

El contenido permitido dentro de <table> es el siguiente **en este orden**:

- Cero o un elemento <caption>: Permite especificar un título para resumir el contenido de la tabla. Permite aumentar la accesibilidad para personas con visibilidad reducida pues permite decidir si el contenido de la tabla es relevante o no para, por ejemplo, utilizar lectores de pantalla.
- Cero o más elementos <colgroup>: Permite agrupar columnas para aplicarles estilos tipográficos, que veremos cuando veamos CSS. Puede tener cero o más subelementos <col>
- Cero o un elemento <thead>: Define un conjunto de filas o *rows* que forman la cabecera de la tabla. Puede tener cero o n elementos <tr> (table rows o filas de tabla). Normalmente se muestra en negrita.
- Cualquiera de los siguientes elementos:
  - Cero o más elementos <tbody>: Envuelve un conjunto de filas o rows que forman parte del cuerpo de la tabla. Puede tener cero o n elementos <tr>. Si se usa <thead>, debe usarse <tbody>. Para conocer más restricciones sobre el uso de <tbody>, visitad *usage notes* de <tbody>
  - Uno o más elemenots <tr>: *table rows* o filas de tabla
- Cero o un elemento <tfoot>: Define un conjunto de filas que resumen el contenido de la tabla. Puede tener cero o más elementos <tr>.



Los elementos <tr> que definen las filas de las tablas pueden contener elementos:
<td>: *Table data cell* o celdas de datos de la tabla
<th>: *Table header cell* o celdas de cabeceras de tabla

Tanto <td> como <th> permiten usar los atributos:

- colspan: Su valor será un número no negativo que indica el **número de columnas que ocupará la celda**. Por defecto, su valor es 1.
- rowspan: Su valor será un número no negativo que indica el **número de filas que ocupará la celda**. Por defecto, su valor es 1.

Por ejemplo, la tabla de la imagen anterior es el resultado del siguiente marcado:

```
<table>
	<caption>Ejemplo de &lt;table&gt;</caption>
	<thead>
		<tr>
			<th colspan="2">Cabecera de la tabla</th>
		</tr>
		<tr>
			<th>Cabecera fila 2, celda 1</th>
			<th>Cabecera fila 2, celda 2</th>
		</tr>
	</thead>
	<tbody>
		<tr>
			<td>Celda con datos 1</td>
```

```
            <td>Celda con datos 2</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td colspan="2">
                Este es el pie de tabla
            </td>
        </tr>
    </tfoot>
</table>
```

La división de las distintas partes de la tabla permite organizar la tabla para su impresión, por ejemplo, repitiendo la cabecera y el pie de tabla en todas las páginas.

Por defecto, los navegadores no muestran los bordes en las tablas. Para mostrarlos, es necesario utilizar propiedades CSS que aún no hemos visto, pero que adelantaremos para mostrar los ejemplos. Por el momento utilizaremos hojas de estilo internas añadiendo el siguiente marcado dentro de la etiqueta <head>, justo antes de </head>

```

<style>
    /*Para mostrar un borde de un píxel negro continuo alrededor de cada tabla, cabecera y celda de datos
    */
    table,
    th,
    td {
        border: 1px solid black;
        border-collapse: collapse; /* Y un único borde unificado */
    }

    /* Para colocar el título bajo la tabla:
    caption{
        caption-side: bottom;
    } */
</style>
```

6.5. Elementos inline vs elementos en bloque

Hasta la versión HTML5, había una división binaria de elementos según el modo en que ocupan el espacio disponible en la página:

- **Elementos en línea o *inline elements*.** Sólo ocupan el espacio necesario para mostrar sus contenidos. Su contenido puede ser texto u otros elementos en línea. Ejemplos:
  - <a>
  - <code>
  - <img>
  - : únicamente agrupa contenido para aplicarle posteriormente algún estilo mediante propiedades CSS.

Una **lista de los elementos en línea se puede encontrar aquí.**

- **Elementos de bloque.** Los elementos de bloque siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea, aunque sus contenidos no lleguen hasta allí. Su contenido puede ser texto, elementos en línea u otros elementos de bloque. Una lista de los elementos de bloque se puede encontrar aquí. Ejemplos:
  - <h1>, <h2>,... <h6>
  - <dl>, <ol>, <ul>, <li>, <dt>, <dd>
  - <p>
  - <table>
  - : únicamente agrupa contenido para aplicarle posteriormente algún estilo mediante propiedades CSS.

Una **lista de los elementos de bloque se puede encontrar aquí.**

Con HTML5, la distinción de categorías de contenido se ha ampliado. Se puede consultar aquí

Copia y pega el siguiente marcado en una nueva página HTML y observa cómo se comporta cada uno de los elementos según su tipo:

```
<h1>Inline elements</h1>
<h2>Texto dentro de <code>&lt;span&gt;</code>></h2>
<p>Aquí hay un <span style="color:red"> span</span> </p>
<h2>Imágenes inline</h2>
<img src="https://images.pexels.com/photos/55814/leo-animal-savannah-lioness-55814.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"
    alt="Una leona rugiendo">
<img src="https://images.pexels.com/photos/235674/pexels-photo-235674.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"
    alt="Perfil de un jaguar">
<h1>Block elements</h1>
<h2>Imágenes anidadas en elementos <code>&lt;div&gt;</code></h2>
<div>
    <img src="https://images.pexels.com/photos/55814/leo-animal-savannah-lioness-55814.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"
        alt="Una leona rugiendo">
</div>
<div>
    <img src="https://images.pexels.com/photos/235674/pexels-photo-235674.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"
        alt="Perfil de un jaguar">
</div>
```

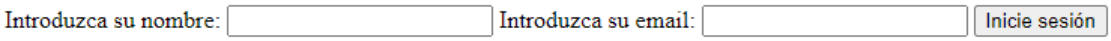
6.6. Formularios

Formularios

Recursos:

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form

Los formularios permiten recoger información por parte del usuario para ser enviada a un servidor utilizando el protocolo HTTP (o su versión segura HTTPS).



HTTP

HTTP es un protocolo basado en la arquitectura **cliente-servidor**: el cliente, normalmente un navegador web, envía una petición HTTP y esta será recibida por otra entidad, el servidor que enviará una respuesta HTTP.
Para poder mostrar una página Web, el navegador **envía una petición** de documento HTML al servidor. Este **responde** con el documento HTML, pero habitualmente un documento HTML involucra otros recursos como scripts, hojas de estilo (CSS), y otros datos que necesite (normalmente vídeos y/o imágenes). Entonces el navegador procesa este documento, y envía más peticiones para solicitar esos recursos relacionados. El navegador, une todos estos recursos y compone el resultado final: la página Web que finalmente muestra al usuario.
Se puede encontrar una imagen que resume este proceso en la primera de las imágenes de esta página.

Para saber más sobre la estructura de los mensajes HTTP se puede consultar este recurso.

Frontend y backend

Una aplicación web está compuesta, a grandes rasgos, de dos partes:

- ***frontend***: una parte visible al usuario y con la que este interactúa. Se ejecuta en el navegador. Se utilizan principalmente las tecnologías HTML, CSS y JavaScript y derivados de estas.
- ***backend***: la parte que se ejecuta en el servidor, que suele tener acceso a bases de datos u otros servicios web. En el lado servidor se pueden encontrar tecnologías como PHP, Python, Java o ASP.NET.

6.7. Herramientas de desarrollo del lado servidor

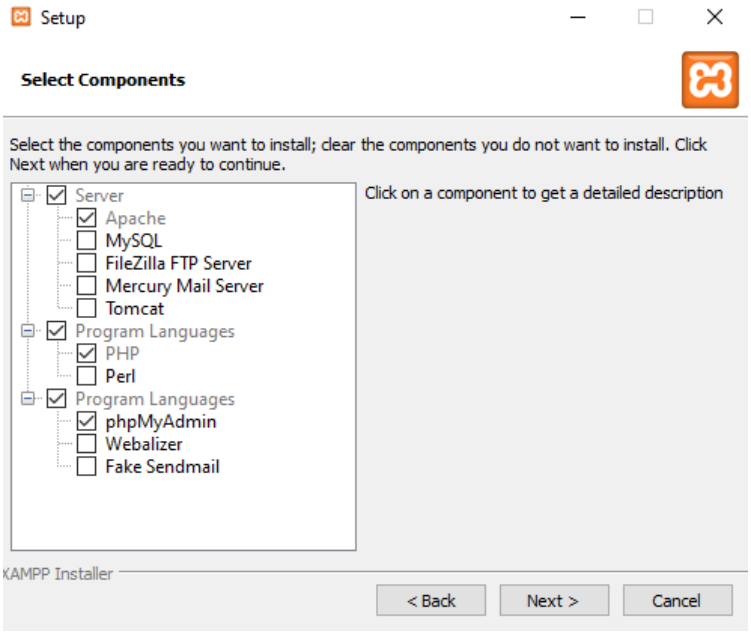
Aunque en este módulo solo veremos la parte *frontend* (con la que interactúa el usuario) de una aplicación web, instalaremos una aplicación ligera de XAMPP para ver cómo se recibirían los datos de un formulario en el lado servidor de una aplicación web realizada con PHP.

Instalación de XAMPP

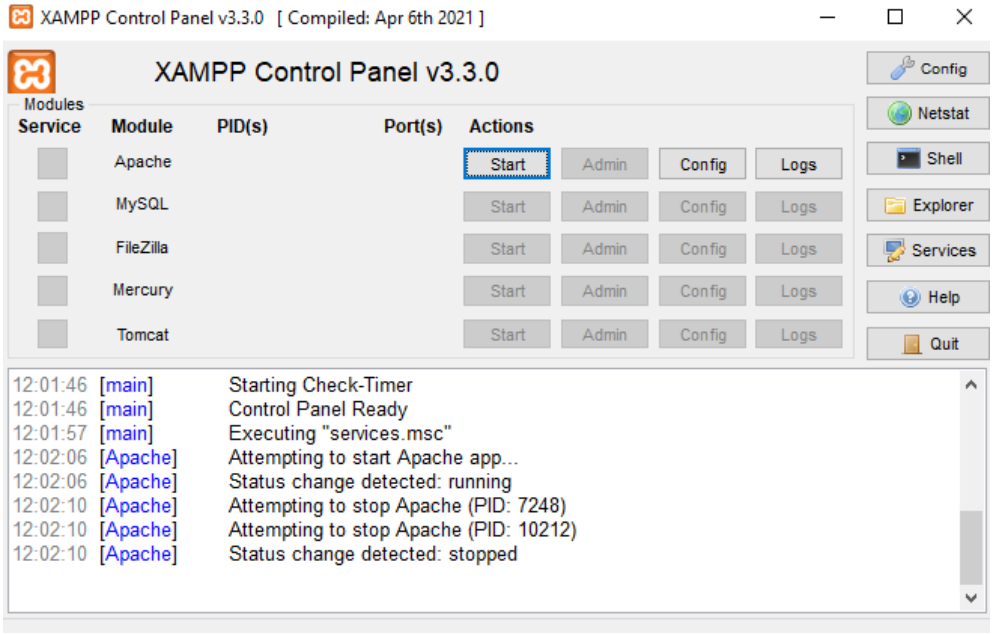
XAMPP es una distribución de un paquete de funcionalidades de software libre que permite incluir un servidor web Apache, un servidor de base de datos MariaDB/MySQL e intérpretes de lenguajes de programación para PHP y Perl. La X indica que es independiente del sistema operativo.

Para realizar la descarga del instalador acudimos a la URL: https://www.apachefriends.org/es/download.html

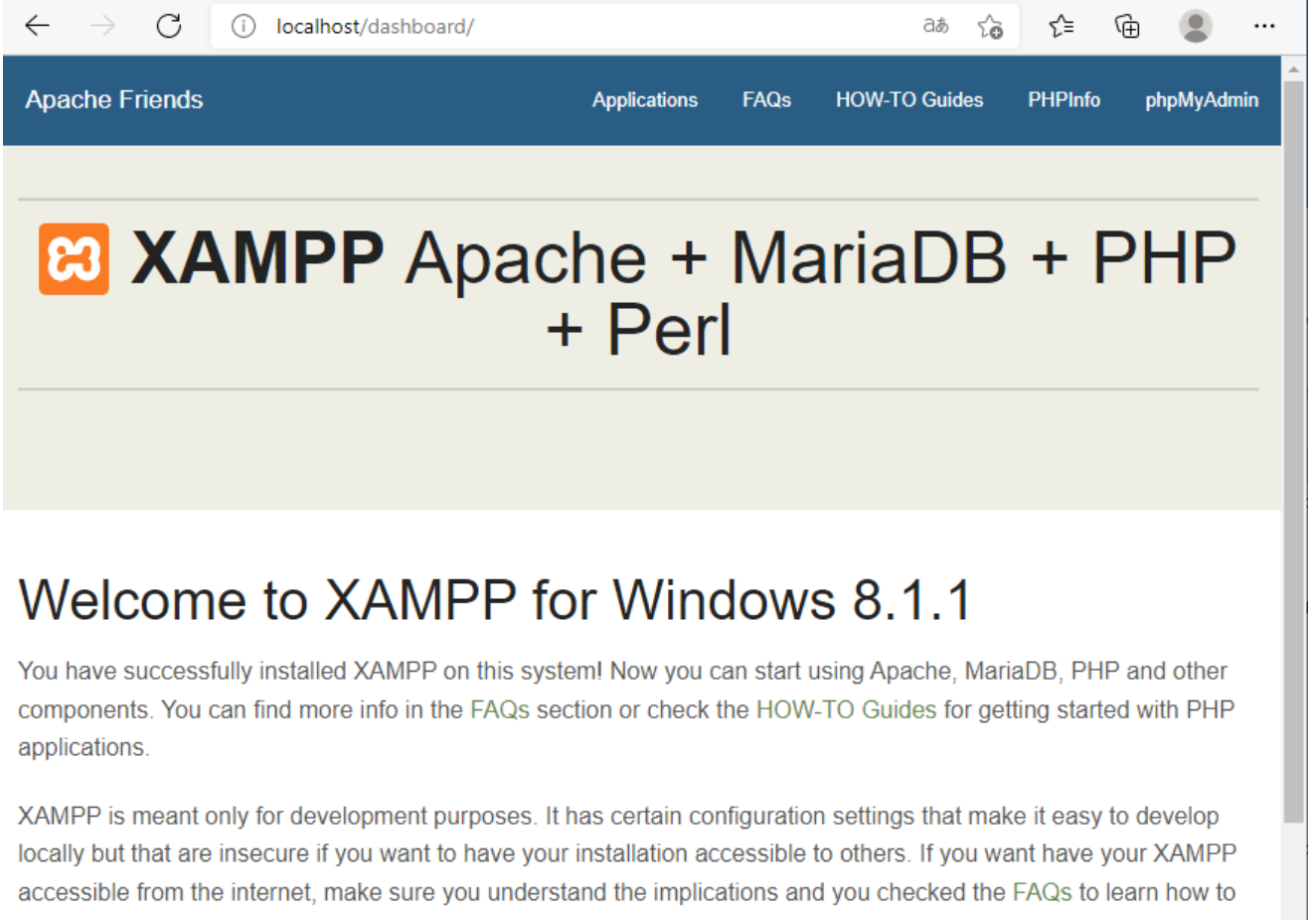
Realizaremos una instalación por defecto. Como nosotros solo vamos a ver cómo llegan los datos al lado del servidor web, nos bastará con incluir el servidor Apache, PHP y phpMyAdmin y eliminaremos la selección de los restantes componentes.



Una vez instalado, iniciaremos el panel de control de XAMPP y arrancaremos Apache presionando el botón Start.



Para comprobar si la instalación y el arranque fue exitoso, visitamos en el navegador la URL: http://localhost:80 y deberíamos ver una página similar a la siguiente:



Con una instalación por defecto en Windows, XAMPP se habrá instalado en C:\xampp
El directorio principal para los documentos WWW es \\xampp\\htdocs.
Si se pone un fichero "test.html" en este directorio, se podrá acceder a él con la URI "http://localhost/test.html".

Nosotros vamos a **crear una carpeta llamada *ud4* dentro de C:\xampp\htdocs** y en esa ubicación vamos a crear un fichero llamado ***formulario.php*** con el siguiente script:

```
<?php

echo "<h1>Estos son los datos recibidos de tu formulario:</h1>";

foreach ($_REQUEST as $clave => $valor) {
    echo "<strong>$clave</strong>:";
    if (!is_array($valor)) {
        echo " $valor";
    } else {
        echo var_dump($valor);
    }
    echo "<br/>";
}
```

Lo que está haciendo este script es escribir en la respuesta un encabezando h1.
A continuación recorre los elementos de un array asociativo (una lista de datos ordenados con pares de clave-valor) que contendrá los elementos enviados por los métodos POST o GET. Para cada par:

- Escribe en la respuesta la clave entre etiquetas
- Si el valor no es un array, lo añade a la respuesta.
- Si el dato recuperado es a su vez un array (un lista de datos ordenados), se escribirá en la respuesta recursivamente.
- Finalmente se añade un salto de línea.

Cuando el navegador reciba la respuesta, añadirá las etiquetas <html> <head> y <body> necesarias para mostrar el contenido de la página.

Si el formulario incluye controles que permitan adjuntar ficheros con **<input type="file">**, deberemos añadir al script lo siguiente:

```
foreach ($_FILES as $input => $infoArr) { //$input será el valor de name en el marcado HTML (sin corchetes)
    if (is_array($infoArr["name"])){ //Si se envía un array de ficheros con el valor de name  en <input type="file"> terminado en []
        foreach($infoArr["name"] as $i=>$value) {
            echo "<strong>File name ".++$i.  " </strong>:";
            echo $value."<br>";
        }
    }
    else{ //Si se envía un único fichero (El valor del atributo name en <input type="file"> no termina con [])
        echo "<strong>File name</strong>: ";
        echo $infoArr["name"]."<br>";
    }
}
```

El script al completo se puede descargar en el apartado de **Recursos** formulario.php

6.8. Formularios: La etiqueta <form>

Recursos:

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form

Principales atributos de <form>

Para realizar el envío de un formulario, <form> deben conocerse al menos los siguientes atributos:

- **action**: La URL de la página o del recurso que procesará **en el servidor** la información enviada. Es un atributo opcional: Si está presente, será la URL de un fichero en el servidor. Si no lo está, se supondrá que será la propia página donde se encuentra la etiqueta <form> en el lado del servidor. También es posible intentar que el navegador utilice una aplicación instalada en la máquina para enviar un correo electrónico mediante `action="mailto:someone@example.com"`. **No es una opción recomendable** por términos de seguridad.

- **method**: Indica el método del protocolo HTTP que se usará: Normalmente se usan:
  - ***GET***: Es un método de HTTP que permite recuperar un recurso o información del servidor. Los datos enviados aparecen en la URL del navegador, por lo que no se debe usar para enviar información sensible. Es el valor por defecto.
  - ***POST***: Es un método de HTTP que permite enviar información al servidor y que causa un cambio en el estado del servidor. Por ejemplo, crear un nuevo usuario o enviar un fichero. Los datos son enviados en el cuerpo de la petición HTTP en lugar de en la cabecera de la petición HTTP y no son visibles en la URL.

- **enctype**: Si el atributo method es **POST**, indica el tipo de contenido que se va a enviar a través de Media types o tipos de medios. Se trata de una cadena de texto que identifica el tipo de contenido enviado y ayuda a los navegadores a presentar dicho contenido. Por ejemplo, **su valor debe ser** `multipart/form-data` si el formulario tiene un <input type="file"> (un control para enviar el contenido de un fichero). Por defecto, su valor es la cadena de texto `application/x-www-form-urlencoded` y para realizar depuraciones (ver qué se está enviando) también se puede usar `text/plain.`

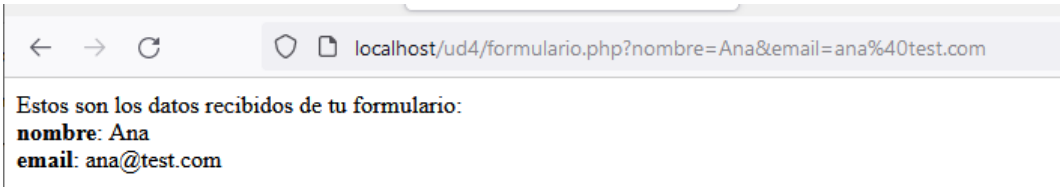
- **name**: En caso de utilizar más de un formulario en un mismo documento HTML, debe ser único en el documento HTML y no puede ser una cadena vacía. Identificará al formulario **en el lado del servidor.**

Ejemplos de formularios con GET y POST

El siguiente marcado HTML vemos 2 formularios: uno con el método GET y otro con el método POST:

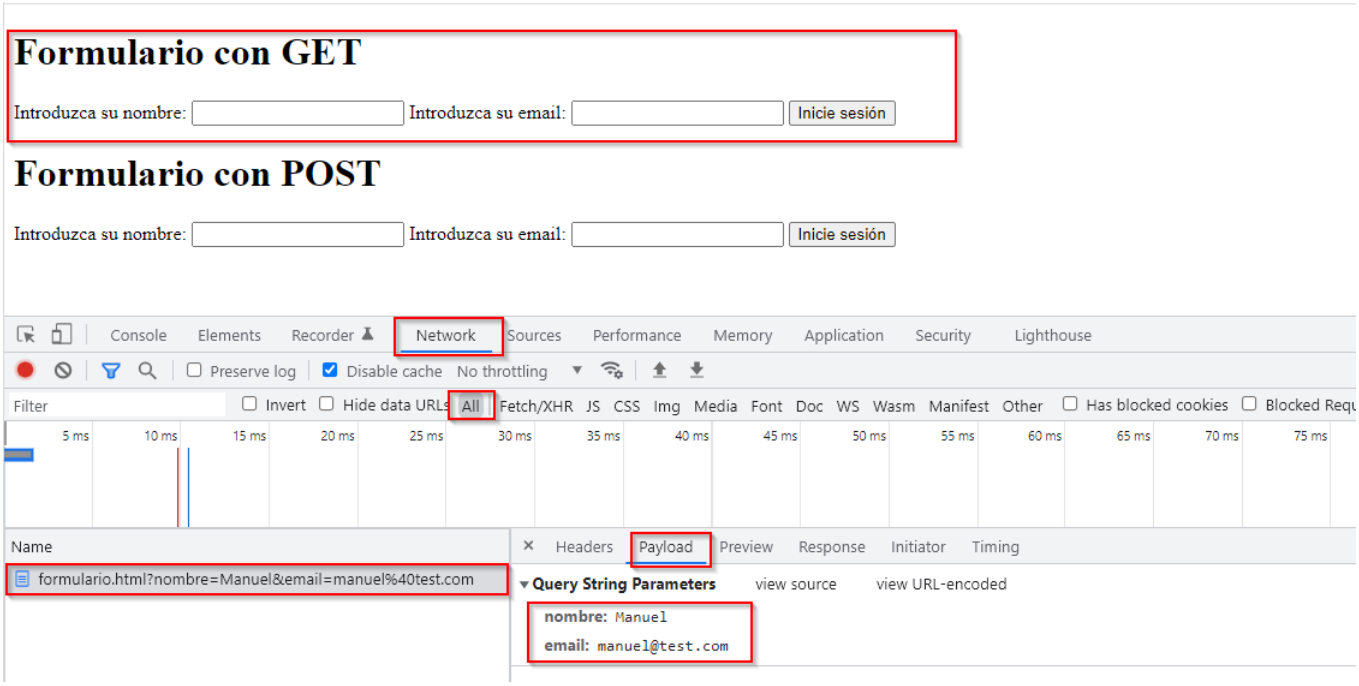
```
    <h1>Formulario con GET</h1>
    <form method="get" action="http://localhost/ud4/formulario.php" name="formulario-get">
        <label for="nombre">Introduzca su nombre: </label>
        <input type="text" name="nombre" id="nombre" required="required">
        <label for="email">Introduzca su email: </label>
        <input type="email" name="email" id="email" required="required">
        <input type="submit" value="Inicie sesión">
    </form>

    <h1>Formulario con POST</h1>
    <form method="post" action="http://localhost/ud4/formulario.php" name="formulario-post">
        <label for="nombre-post">Introduzca su nombre: </label>
        <input type="text" name="nombre-post" id="nombre-post" required="required">
        <label for="email">Introduzca su email: </label>
        <input type="email" name="email-post" id="email-post" required="required">
        <input type="submit" value="Inicie sesión">
    </form>
```

En el caso del formulario con método **GET**, veremos que tras hacer clic en el botón de Inicio de sesión, **la URL nos mostrará** algo similar a lo siguiente:



- Después de la URL que procesará el formulario, formulario.php en nuestro ejemplo, aparece **el símbolo ?** y a continuación siguen los pares de *nombre=valor*
- El *nombre* viene dado por el valor del atributo *name* de los elementos de entrada *input* que veremos en próximas secciones
- Cada par *nombre=valor* se separa del siguiente mediante **el símbolo &**
- Algunos caracteres que no son válidos en una URI deben ser codificados, es decir, reemplazados por otros. Es el caso de **@** que se reemplaza por **%40**.

Esto mismo se puede observar en el navegador **Chrome**, con las herramientas de desarrollador o pulsando **F12**, en la pestaña de **Network**.
Tras introducir los datos en el formulario con GET y pulsar el botón de Inicie sesión, podremos ver las peticiones HTTP que se generan, su método GET y si hacemos clic sobre la URL en la columna Name, veremos datos de las cabeceras HTTP y su carga de datos o *payload*, que en este caso forman parte de la URL.



Sin embargo, no se verá reflejado en la URL si usamos el método POST:

Ejemplo de formulario con mailto

```
<h1>Formulario con mailto</h1>
```

```
   <form action="mailto:someone@example.com" method="post" enctype="text/plain">
```

```
       <label for="nombre-post">Introduzca su nombre: </label>
```

```
       <input type="text" name="nombre-post" id="nombre-post"  required="required">
```

```
       <label for="email">Introduzca su email: </label>
```

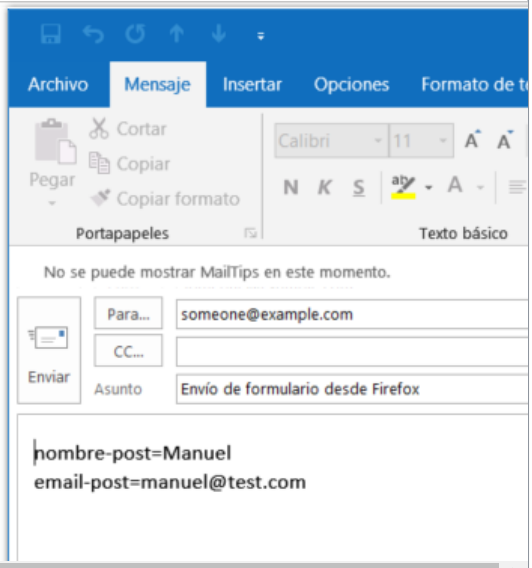
```
       <input type="email" name="email-post" id="email-post" required="required">
```

```
       <input type="submit" value="Enviar">
```

```
   </form>
```

Formulario con mailto

Introduzca su nombre: Manuel Introduzca su email: manuel@test.com Enviar

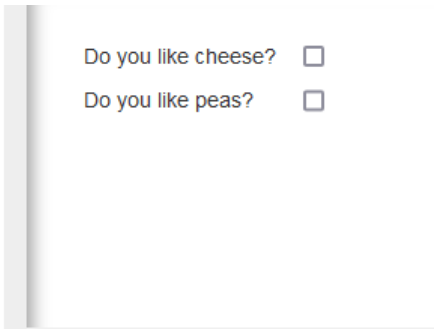


Elementos de formularios

Elementos <label>

Los elementos <label> ayudan a identificar la información que se espera como entrada en un control <input>.
Deben asociarse por medio del atributo ***for*** de forma que su valor sea el valor del atributo ***id*** de la etiqueta <input> relacionada.

```
<div class="preference">
    <label for="cheese">Do you like cheese?</label>
    <input type="checkbox" name="cheesebox" id="cheese">
</div>

<div class="preference">
    <label for="peas">Do you like peas?</label>
    <input type="checkbox" name="peasbox" id="peas">
</div>
```



Esto favorece la accesibilidad pues las aplicaciones lectoras de pantalla pueden indicar fácilmente el tipo de dato esperado en cada control.
Por otra parte, haciendo clic sobre la propia etiqueta, se consigue el mismo efecto que haciendo clic sobre el control <input>.

Otra alternativa sería anidar el control <input> dentro de <label>. El resultado sería el mismo sin necesidad de usar el atributo for.

```
<label>Do you like peas?
  <input type="checkbox" name="peas">
</label>
```

Elementos <input>

Permiten al usuario introducir información. Existen multitud de tipos de elementos <input>y estos se especifican con el atributo ***type***. Si el atributo **type** no está presente, su valor por defecto es ***text***. Se trata de un elemento vacío o ***void element*** y no debe tener etiqueta de cierre.

Algunos tipos de elementos <input> son:

<input type="text">: Es el valor por defecto para el atributo *type*. Muestra una caja de texto que permite introducir texto. Cuenta con atributos para establecer la longitud máxima y mínima permitidas, un patrón con expresiones regulares para limitar las posibles entradas, el ancho de la caja de texto, si es de solo lectura, un placeholder o una pista del texto esperado, etc.

<input type="submit">: Muestra un botón que, una vez que el usuario haga clic sobre el mismo, enviará el contenido del formulario al servidor. Si existe un único elemento <button> dentro de un form, este será tratado como <input type="submit">.

<input type="reset">: Muestra un botón que, una vez presionado, elimina el contenido de los controles de un formulario. No está recomendado porque puede resultar frustrante para el usuario si lo pulsa por error.

<input type="button">: Muestra un botón genérico sin funcionalidad por defecto. Ha de añadírsele funcionalidad mediante JavaScript, un lenguaje de programación del lado cliente que se ejecuta en el navegador. Se recomienda el uso de el elemento <button> en su lugar porque permite incluir HTML o incluso imágenes en su interior.

Los diferentes tipos de <input> y sus atributos se pueden consultar aquí.

Atributos de todos los elementos <input>:

- **name**: El nombre con el que se recibirá el dato en el servidor. Debe ser único en el formulario y **debe estar presente para poder identificarlo en el servidor**. En PHP, en el caso de que se permitan enviar múltiples valores, como en una lista desplegable de selección múltiple o donde se permitan adjuntar múltiples ficheros, el nombre deberá terminar con corchetes para que ser reciba un array en el servidor:
  `<input type="file" name="ficheros[]" id="fichero" multiple>`
- **value**: El valor inicial del control. Puede ser el texto de un type="button".
- **disabled:** Indica si el control debe estar deshabilitado. Es opcional.
- **autocomplete**: Permite que se muestren opciones para completar el carmpo.
- **autofocus:** Sitúa el foco de la página en ese control automáticamente tras la carga de la página inicial. El **foco** se refiere a **qué** control en la pantalla recibe actualmente la entrada desde el teclado. Por ejemplo, si es un campo de texto, se verá el cursor parpadeante en el campo de texto.

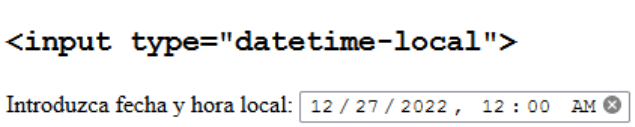
Atributos de casi todos los elementos <input>

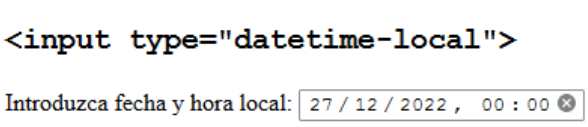
- **readonly**: Indica que el valor no es editable
- **required**: Indica que el valor no debe estar vacío antes de ser enviado al servidor.

Cómo cambiar el idioma de visualización en Firefox

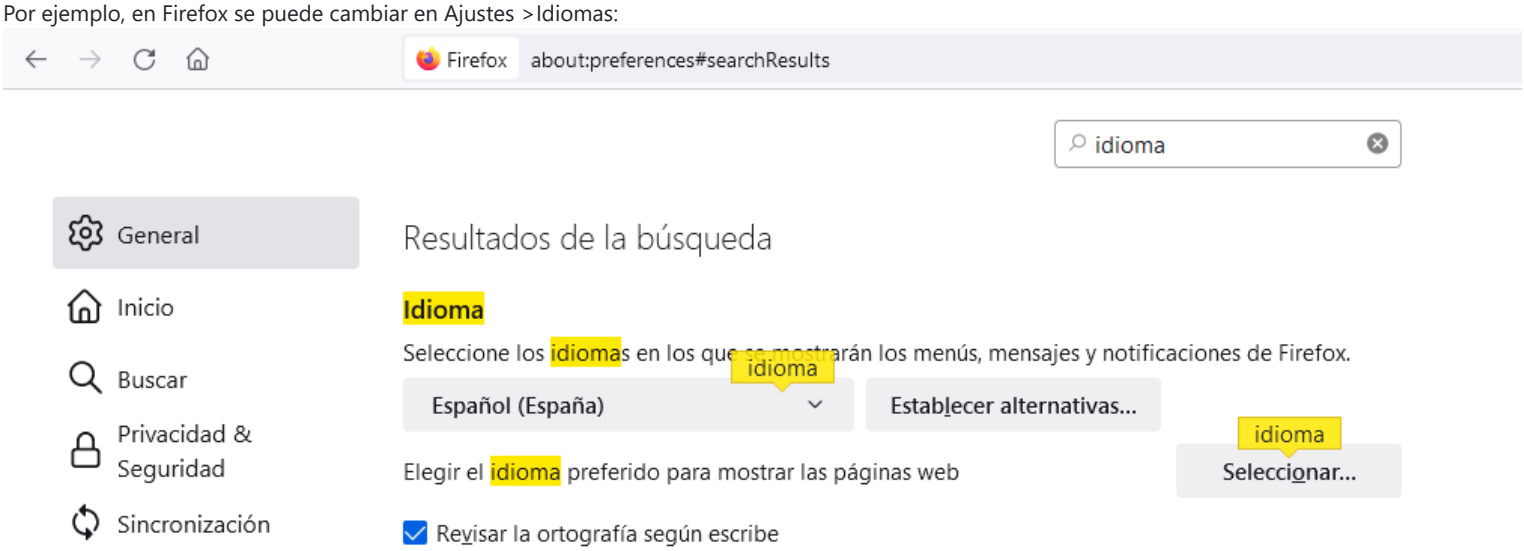
Para mostrar las fechas, el navegador tiene en cuenta el idioma de visualización preferido.

En las siguientes imágenes se observa cómo cambia el formato de fecha con el inglés de EE.UU. (a la izquierda) con respecto al español de España:





Por ejemplo, en Firefox se puede cambiar en Ajustes >Idiomas:



Elemento <datalist>

Permite definir una lista de opciones disponibles o sugerencias para los valores de un elemento <input>. No obliga a escoger entre sus valores y el usuario puede introducir otro valor no contemplado en la lista. Se puede aplicar a los <input> de tipo `text`, `search`, `url`, `tel`, `email`, `date`, `month`, `week`, `time`, `datetime-local`, `number`, `range` y `color`.

El elemento **<datalist>** debe tener un atributo id.

El elemento **<input>** debe tener el atributo list cuyo valor debe ser el valor del atributo id de **<datalist>**.

```
<label for="ice-cream-choice">Choose a flavor:</label>
<input list="ice-cream-flavors" id="ice-cream-choice" name="ice-cream-choice" />

<datalist id="ice-cream-flavors">
    <option value="Chocolate">
    <option value="Coconut">
    <option value="Mint">
    <option value="Strawberry">
    <option value="Vanilla">
</datalist>

```

Elemento <select>

Permite crear una lista desplegable donde es posible seleccionar una o más opciones. Cada opción ha de incluirse en un elemento <option>. Los elementos <option> se pueden agrupar en un grupo con la etiqueta <optgroup>

Elemento <textarea>

Crea una caja de texto donde el usuario puede introducir texto de más de una línea. Cuenta, entre otros con los atributos rows y cols para indicar el número de filas y anchura media de carácter va a ocupar.

7. Elementos semánticos

Se consideran **elementos semánticos** aquellos que, simplemente con su nombre, dan una idea del contenido que engloban: Algunos ejemplos de elementos que hemos visto hasta el momento son: <p>, encabezados: <h1>, <h2>, ... <h6>, <table>, <img>, <form>,etc.

Sin embargo, ** y no son elementos semánticos** pues no nos hacen suponer nada sobre su contenido.

7.1. HTML5: Estructura semántica en una página

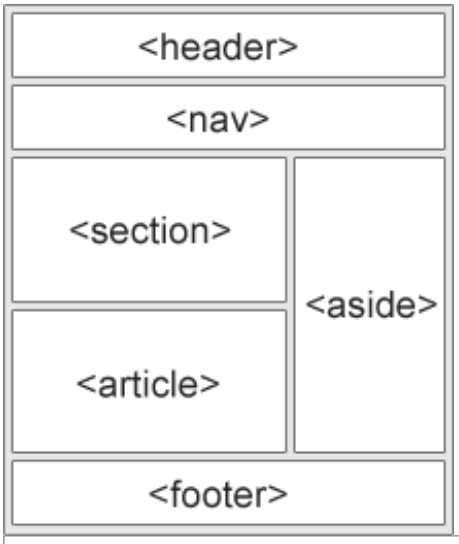
Etiquetas de estructura semántica en una página HTML5

HTML5 introdujo etiquetas que permiten estructurar el contenido de la página HTML e indican, a grandes rasgos, **cuál es la naturaleza de su contenido, de ahí que se consideren etiquetas semánticas**. De esta forma permiten entender más fácilmente la estructura de la página y contribuyen a la accesibilidad de la página HTML para los lectores de pantalla.

Las etiquetas de **seccionado** que permiten estructurar de forma semántica una página HTML son:

- `<header>`: Puede definir la **cabecera de la página o la cabecera de una `<section>` o `<article>`**. Suele contener uno o varios encabezados (<h1>, <h2>, etc.), logos, autor, sección de navegación, etc.
- `<main>`: encierra el contenido principal de la página. Solo se usará **una vez por página** directamente dentro del elemento `<body>`.
- `<nav>`: agrupa los enlaces de **navegación principales** de la página (no es necesario que siempre que haya enlaces se use `<nav>`). Podría aparecer más de una vez en la página.
- `<article>`: Enmarca un bloque de **contenido que tiene sentido por sí mismo**. Por ejemplo, una entrada en un blog o un artículo en un periódico.
- `<section>`: Delimita una **sección genérica de un documento**. Una buena práctica es que comiencen con su propio encabezado. Normalmente está pensado para contenidos dependientes, pero diferenciados. Por ejemplo, un <article> que contenga varios elementos <section>. Sin embargo, podría existir una <section> que contenga varios <article> relacionados por la misma temática.
- `<aside>`: incluye contenido que no está directamente relacionado con el contenido principal, pero que puede aportar **información adicional** (publicidad, resúmenes, enlaces relacionados, etc.).
- `<footer>` representa contenido al **final de una página o de una sección**. Suele incluir información acerca de el autor de la sección, datos de derechos de autor o enlaces a documentos relacionados

Una posible representación gráfica de esta estructura podría ser la siguiente:



Un ejemplo de marcado HTML con este tipo de etiquetas podría ser el siguiente:

```
<header>
    <h1>¡Despierta Vigo!</h1>
    <p><a href="news.html">Noticias</a> -
       <a href="blog.html">Blog</a> -
       <a href="forums.html">Foros</a></p>
    <p>Última modificación: <span>2009-04-01</span></p>
    <nav>
     <h2>Navegación</h2>
     <ul>
      <li><a href="articles.html">Índice de todos los artículos</a></li>
      <li><a href="today.html">Noticias de hoy</a></li>
```

```
      <li><a href="successes.html">Nuestros casos de éxito</a></li>
     </ul>
    </nav>
</header>
<main>
    <article>
     <header>
      <h2>Hoy en Vigo</h2>
     </header>
     <section>
      <h3>Conciertos</h3>
      <p>Cosas sobre conciertos</p>
     </section>
     <section>
       <h3>Tráfico</h3>
       <p>Cosas sobre tráfico</p>
      </section>
     <footer>
      <p>Posted <time datetime="2009-10-10">Thursday</time>.</p>
     </footer>
    </article>
 ... más artículos ...
</main>
<aside>
       Entradas ya  a la venta en <a href="https://www.ticketmaster.es/">Ticketmaster</a>
</aside>
<footer>
    <p>Copyright ©
     <span>2010</span>
     <span>The Example Company</span>
    </p>
    <p><a href="about.html">Sobre nosotros</a> -
       <a href="policy.html">Política de privacidad</a> -
       <a href="contact.html">Contacto</a></p>
</footer>
```

Recursos

Se pueden consultar los siguientes recursos para más información:

- https://developer.mozilla.org/es/docs/Learn/HTML/Introduction_to_HTML/Document_and_website_structure#elementos_de_dise%C3%B1o_html_en_detalle
- https://www.w3schools.com/html/html5_semantic_elements.asp

Se pueden encontrar más ejemplos de la propia documentación de HTML5 para cada etiqueta aquí

BUENAS PRÁCTICAS

Los elementos `div` no conllevan valor semántico y debemos usarlos solo cuando no encontremos una opción semántica ya existente, reduciendo su uso al mínimo.

7.2. HTML5: Otros nuevos elementos semánticos

Además de los elementos estructurales de una página web, se consideran nuevos elementos semánticos:

<figure>

<figure>: representa contenido independiente, normalmente una imagen, una ilustración, un diagrama, un fragmento de código, o un esquema al que se hace referencia en el texto principal. Se le puede asociar un título con el elemento <figcaption> en su interior (como el primero o el último hijo)

```
<figure>
   <img src="https://images.pexels.com/photos/55814/leo-animal-savannah-lioness-55814.jpeg?
auto=compress&cs=tinysrgb&dpr=1&w=500"
       alt="Una leona rugiendo">
   <figcaption>Una leona rugiendo</figcaption>
</figure>
```

```
   <figure>
     <code>
          <pre>
          &lt;?php
          echo "&lt;h1>Estos son los datos recibidos de tu formulario:&lt;/h1>";
          foreach ($_REQUEST as $clave => $valor) {
             echo "&lt;strong>$clave&lt;/strong>:";
             if (!is_array($valor)) {
                echo " $valor";
             } else {
                echo var_dump($valor);
             }
             echo "&lt;br/>";
          }
          </pre>
     </code>

   <figcaption>Script en PHP que muestra los datos recibidos en el lado servidor mediante GET o POST a través de un formulario
HTML</figcaption>
  </figure>
```

<mark>

<mark> resalta un texto por su especial relevancia, normalmente con color de fondo amarillo

```
<p> <mark><a href="https://developer.mozilla.org/es/docs/Web/HTML/Element/mark">Aquí</a></mark> se puede consultar un ejemplo.</p>
```

<details>

<details> permite añadir un contenido resumido con un control, normalmente un triángulo que rota si se hace clic sobre él mostrando contenido más detallado. El contenido resumido debe anidarse en la etiqueta<summary>. Internet Explorer no lo soporta.

```
   <details>
```

```
       <summary>Resumen de información desplegable</summary>
```

```
       <p>Más información se puede añadir aquí</p>
```

```
   </details>
```

8. Elementos multimedia

La utilización de contenido multimedia consiste en utilizar distintos medios de expresión y presentación de la información, como texto, imágenes, audios, vídeos o animaciones.

<video>

La etiqueta **<video>** permite incluir un vídeo en un documento HTML. Se consideran atributos relevantes los siguientes:

- **controls**: Si está presente, muestra controles para iniciar, pausar, parar la reproducción y controlar el volumen.
- **autoplay**: Su valor puede ser true o false. Indica si debe empezar a reproducirse el vídeo tan pronto como sea posible aunque no se haya cargado completamente su contenido. No está recomendado pues puede resultar una experiencia desagradable para el usuario. En algunos navegadores no funcionará hasta que se establezca muted. Para que no se reproduzca no es suficiente con añadir autoplay="false": Habrá que retirar el atributo autoplay.
- **loop**: Si está presente indica que volverá automáticamente al inicio cuando la reproducción llegue a su fin.
- **muted**: Indica si empezará a reproducirse inicialmente sin sonido. Por defecto, su valor es false.
- **width** y **height**: Indica el número de píxeles que ocupará el área dedicada al vídeo en anchura y altura respectivamente. No necesita la unidad, basta el número.
- **poster**: La URL de la imagen que se mostrará mientras el vídeo no esté disponible antes de su reproducción o hasta que se comience a reproducir.
- **src**: Indica la URL donde se puede encontrar el archivo del vídeo. Es opcional y en su lugar se puede usar <source>
- Es posible añadir un mensaje en caso de que el navegador no soporte la etiqueta <video> justo antes de cerrar </video>

Recursos para encontrar y descargar vídeos para uso no comercial:

- https://pixabay.com/es/videos/
- https://www.pexels.com/es-es/videos/

<source>: Normalmente se utiliza para servir el mismo contenido multimedia en varios formatos para ampliar la posibilidad de soporte bajo diferentes navegadores. **Se puede utilizar dentro de <video>, <audio> y <picture>**. El navegador comprobará por orden de aparición si alguno de los formatos especificados coincide con un formato soportado. Si no, comprueba el siguiente elemento <source>. Si no soporta ninguno, puede que, dependiendo del navegador se muestre un texto o simplemento no sea posible la reproducción. <source> utiliza principalmente 2 atributos:

- **src**: indica la URL donde se encuentra el fichero
- **type**: indica una cadena de texto con el tipo MIME. Por ejemplo: video/ogg, video/mp4 o video/quicktime. Más valores aquí.

```
<video controls>
  <source src="myVideo.webm" type="video/webm">
  <source src="myVideo.mp4" type="video/mp4">
  <p>Su navegador no soporta <code>&lt;video></code></p>
</video>
```

<audio>

La etiqueta <audio> permite incluir un archivo de sonido en la página HTML. El elemento `<audio>` funciona exactamente de la misma forma que el elemento `<video>`, con algunas pequeñas diferencias:

- **No cuenta con width** y **height** pues no hay contenido visual que mostrar
- **No cuenta con** el atributo **poster**

```
<audio controls>
  <source src="viper.mp3" type="audio/mp3">
  <source src="viper.ogg" type="audio/ogg">
  <p>Su navegador no soporta <code>&lt;audio> </code></p>
</audio>
```

Recursos para encontrar un banco de sonidos gratuito:

- https://www.sshhtt.com/home

<picture>

Permite especificar múltiples elementos `<source>` y un elemento `<img>`.

Los elementos <source> proveen distintas versiones de una imagen para diferentes escenarios de dispositivos, por ejemplo, en función de su tamaño. Esto se puede hacer gracias al atributo **media** de <source> que permite especificar una ***media query (una expresión CSS)*** que el navegador evaluará para seleccionar un elemento `<source>`. Si la media query se evalúa a `false`, el elemento `<source>` es omitido.

Un ejemplo gráfico se puede encontrar aquí con la herramienta de **Try it yourself**

```
<picture>
```

```
  <source media="(min-width: 650px)" srcset="img_food.jpg">
```

```
  <source media="(min-width: 465px)" srcset="img_car.jpg">
```

```
  <img src="img_girl.jpg" style="width:auto;">
```

```
</picture>
```

<iframe>

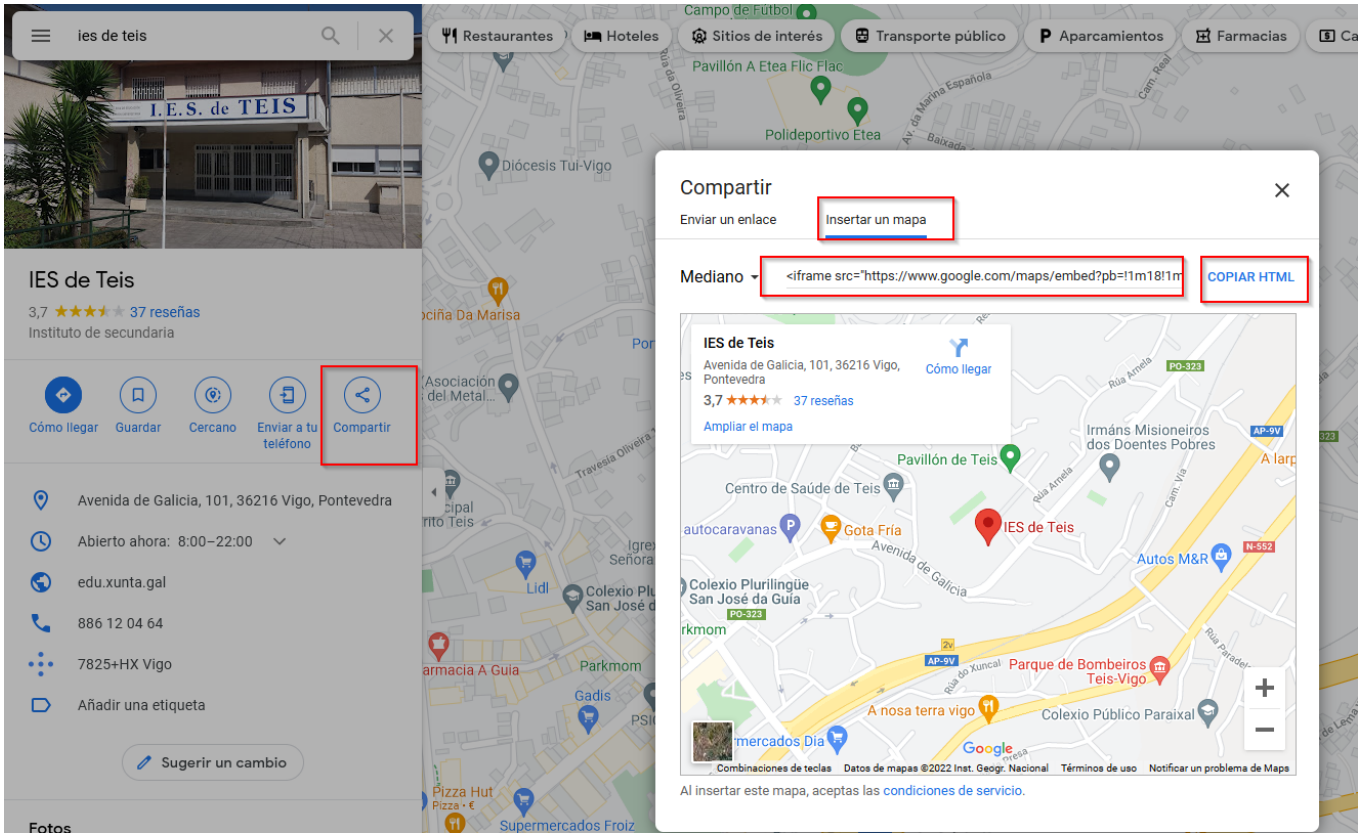
Permite incrustar una página HTML dentro de otra. Se pueden controlar su tamaño con los atributos width y height

```
    <iframe src="https://www.edu.xunta.gal/centros/iesteis/" title="Web del IES de Teis" width="500"
```

```
        height="500"></iframe>
```

Un uso habitual lo encontramos con las ubicaciones de Google Maps. Para ello:

1. Buscamos la ubicación
2. Hacemos clic sobre compartir
3. Para insertar un mapa en una página HTML se nos permite copiar un elemento <iframe> con más parámetros




```
<iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d2953.2710057619124!2d-
8.692259584277991!3d42.251384849949694!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0xd2f62e588cfce69%3A0x378485bfa6edd1be!2sIES%20de%20
 width="600" height="450" style="border:0;" allowfullscreen="" loading="lazy"></iframe>
```

<object>

Representa un recurso externo, que puede ser tratado como una imagen, vídeo, otra página web o como un recurso que debe ser manejado por un plugin.
Han de incluirse por lo menos los atributos:

- data: La URL del recurso
- type: El tipo MIME del recurso.

```
  <object data="files/UD2.1.1-DTD.pdf" type="application/pdf" height="600">
```

Siempre que exista un elemento HTML específico para el recurso es conveniente utilizarlo. Por ejemplo, <video>, <audio>, <img>, <iframe>.