

# LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

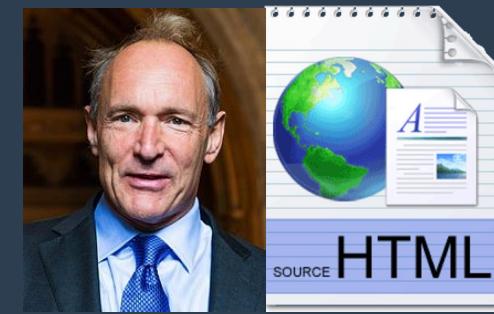


**Unidad 3: Lenguajes de marcas en entornos web**

# ÍNDICE

- **Páginas web**
- **Prototipado web**
- **Control de versiones con GIT**
- **¿Cómo funciona la web?**
- **Estructura HTML**

# Páginas web



- **HTML** (HyperText Markup Language) es el lenguaje de marcas más utilizado para escribir documentos y que estos sean accesibles a través de la web.
- Su impulso se lo debemos a Tim Berners-Lee que fue su diseñador (lo creó en 1989 y se estandarizó en 1993 por la IETF) y que creó la **W3C** (World Wide Web Consortium) que organiza su funcionamiento y reglas de uso.

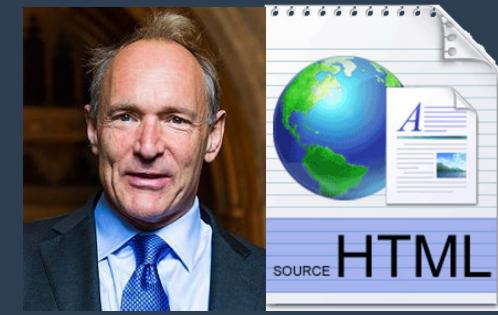


**Vídeos:** Introducción a la evolución de HTML y su programación.

- En su origen, HTML era capaz de proporcionar un mecanismo sencillo, flexible y universal para almacenar y transmitir información entre diferentes sistemas informáticos. Por eso tuvo esa repercusión y se ha expandido su uso.
- Otro elemento fundamental en el éxito de la web son los **navegadores**, ya que permiten procesar HTML y mostrar un resultado al usuario. Recordad que HTML es descriptivo y de presentación mientras que XML sólo es descriptivo. Además, sólo se presentan las reconocidas en HTML, no otras creadas por nosotros.
- Claro está, este diseño sólo constaba con 18 etiquetas y desde entonces se han ido incorporando más hasta el actual HTML5 que tiene unas 142.

**Debate:** ¿Crees que todos los navegadores integran bien HTML5? ([enlace](#))

# Páginas web

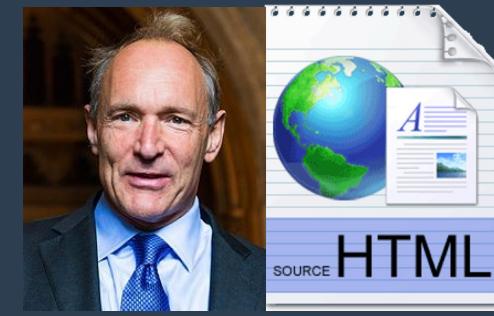


- Así ha ido cambiando HTML a lo largo del tiempo ([web](#)):



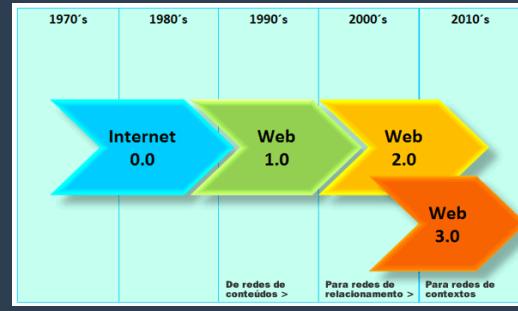
**Vídeo:** ¿Por qué se ha popularizado tanto HTML5?

# Páginas web



- Entre sus avances fundamentales tenemos:
  - El lenguaje cada vez ha ido incorporando nuevas etiquetas más potentes, que permiten incluir en los documentos HTML, tablas, capas, marcos, imágenes, vídeos, sonidos, ...
  - Se han añadido lenguajes de script (especialmente JavaScript) con código incrustado en las propias páginas HTML que permiten añadir funcionalidades extra y dinamismo a las páginas web.
  - Han aparecido lenguajes y tecnologías que permiten ejecutar acciones avanzadas en el servidor que aloja las páginas web. Entre ellas CGI, PHP, ASP o JSP.
  - Se ha incorporado lenguajes de estilo (como CSS) para generar un formato de documento más avanzado y fácil de mantener.
  - Se han añadido utilidades para gestión avanzada de JavaScript con XML (AJAX) para dar aún más interactividad y dinamismo a las páginas.
  - Hubo un tiempo donde se utilizó mucho las páginas con Flash o los applets de Java para dar mayor funcionalidad. Hoy ya se ha abandonado porque sobrecarga mucho la página.
  - Se permiten elementos semánticos para dar significado al contenido (meta tags).
  - Se añaden cada vez más plugins a los navegadores que así son capaces de mostrar imágenes, sonido, vídeo y otros elementos multimedia en las propias páginas.
  - Hay multitud de herramientas de edición y framework que facilitan su desarrollo aunque no tengas conocimientos técnicos. Esto se conoce como edición WYSISWYG Ej: blogspot vs VS CODE

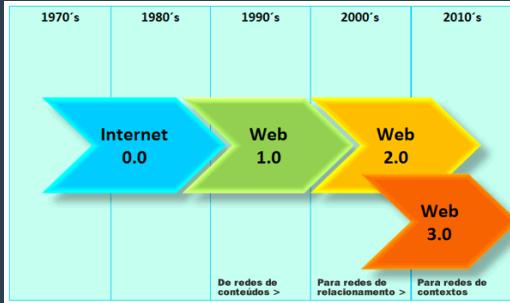
# Páginas web



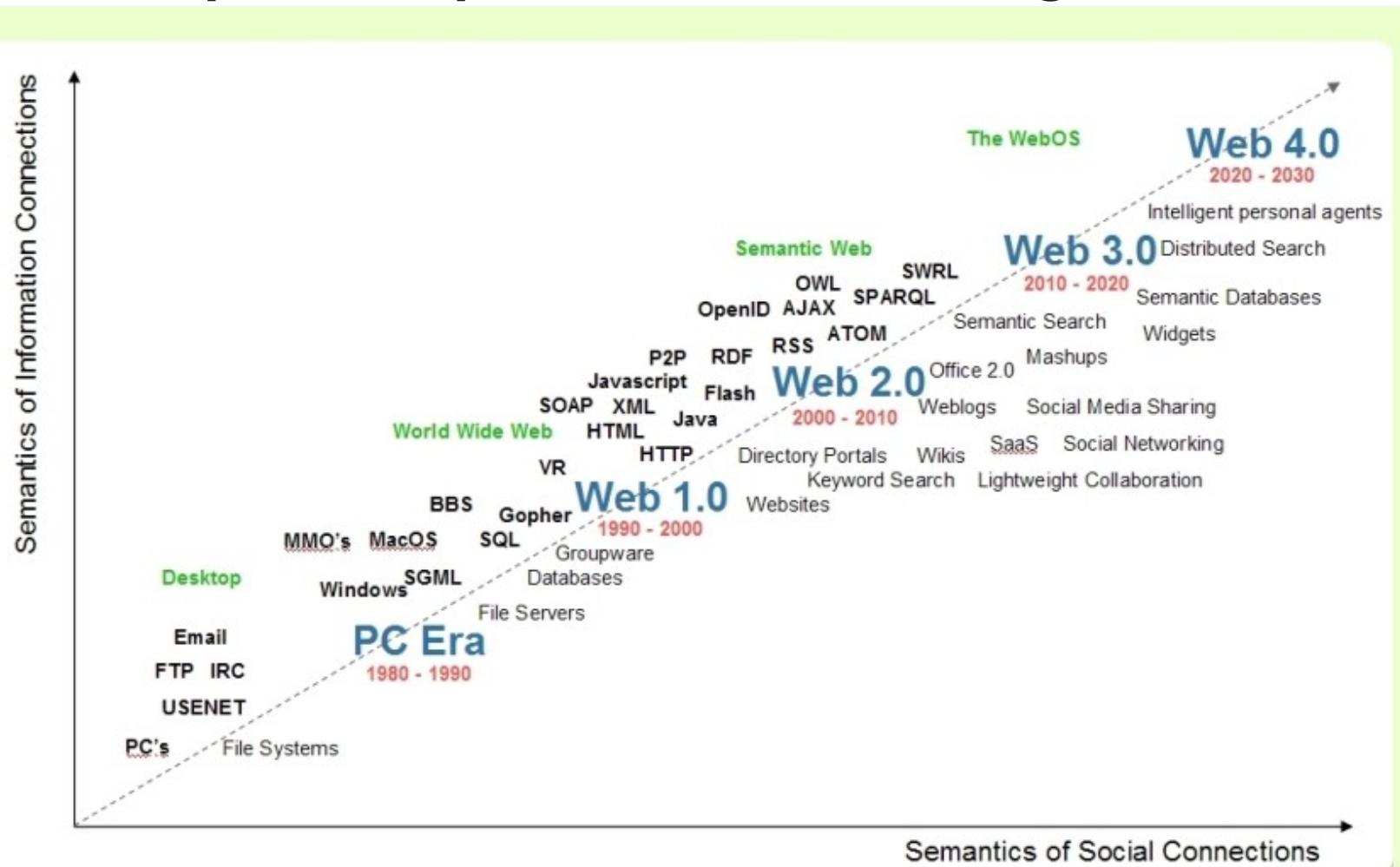
- **Este desarrollo ha dado lugar a diferentes tipos de web:**
  - **Web 1.0 o Web Estática.** Páginas desarrolladas en lenguaje HTML para mostrar información. El usuario tiene un papel pasivo, siendo actualizadas por el administrador.
  - **Web 2.0 o Web Social.** Páginas dinámicas y colaborativas donde el usuario es el protagonista creando, compartiendo, opinando, participando etc. Hoy en día, la mayoría de páginas son de este tipo.
  - **Web 3.0 o Web Semántica.** Estas páginas emplean datos semánticos que facilitan el acceso y la respuesta de información de forma eficiente, rápida, sencilla y especializada. Ejemplos serían Siri o Cortana.
  - **Web 4.0 o Web total:** Páginas en fase de desarrollo que pretenden integrar inteligencia artificial, imágenes en tres dimensiones y lenguaje natural con respecto a las páginas anteriores. Un ejemplo sería realizar la búsqueda en un teléfono inteligente como... “Quiero volar a Sevilla mañana para ver el Alcázar” y que este se encargara de buscarme la mejor opción de llegar a Sevilla, gestionar la reserva del viaje y la entrada al monumento.

**Vídeo:** Web 1.0 a 4.0

# Páginas web



- Y están impulsadas por diferentes tecnologías...



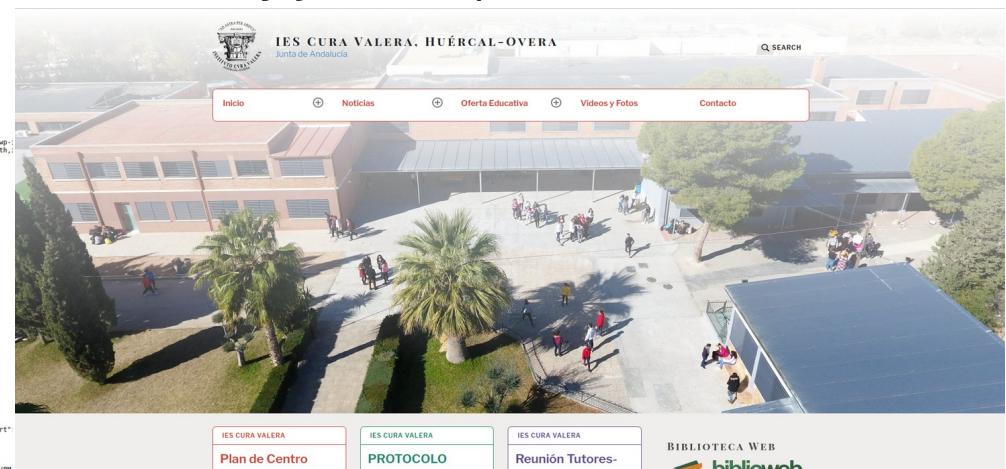
Source: Radar Networks & Nova Spivack, 2007 – [www.radarnetworks.com](http://www.radarnetworks.com)

# Páginas web



- **Los navegadores web**

- Para acceder a las páginas web se utiliza un navegador que se encarga de comunicar el ordenador con los servidores que albergan las páginas web, las descarga y las muestra.
  - Ejemplo: Web del Instituto que incluye HTML, CSS y JavaScript



- **Importante:** Los navegadores tienen la capacidad de interpretar el código HTML de forma más visual y cambiar el tipo de letra, colores, incorporar imágenes, sonido, etc. de mostrar el HTML conforme lo va procesando. El problema es que si encuentra algún error intentará “solucionarlo” y si no es crítico no nos daremos cuenta. Para evitar esto necesitamos **validar** las páginas HTML. (**Vídeo**)
  - ¡Ojo!, muchas de las páginas actuales no pasan esta validación!!

# Páginas web



- **Los navegadores web**

- Hay muchos navegadores disponibles en Internet:

- **El proyecto Chromium:** Es un navegador de código abierto que funciona como base de múltiples navegadores web, como por ejemplo, Google Chrome u Opera. La idea del proyecto es que sea lo más rápido, simple y eficiente posible, permitiendo que otros desarrolladores lo tomen como base, añadiendo nuevas funcionalidades "encima".
- **Google Chrome:** Es el navegador más utilizado en la actualidad, y uno de los navegadores (si no el que más) que implementa con mayor rapidez las últimas novedades tecnológicas dentro de campos como CSS3, HTML5 o JavaScript, entre otros. También es bastante criticado por el alto consumo de recursos, sobre todo en cuanto a memoria RAM, que en equipos con pocos recursos puede ser bastante molesto.



Casi todo el mundo utiliza Chrome en móvil o PC pero no es necesariamente el mejor.

# Páginas web



- **Los navegadores web**

- Hay muchos navegadores disponibles en Internet:
  - **Mozilla Firefox:** Nace de un navegador llamado Mozilla, que a su vez era el resurgir del antiguo navegador Netscape. Se caracteriza por utilizar software libre y es un navegador que gestiona mejor la memoria RAM que Google Chrome.
  - **Internet Explorer:** Internet Explorer era el navegador de Microsoft para competir con Netscape. Su uso se ha debido a que venía integrado en Windows, no porque sea el mejor. Actualmente se está dejando de usar por la versión Edge, aunque sólo se use en Windows 10
  - **Safari:** Es el navegador que viene incluido en los sistemas de Apple. Aunque se trata de un navegador muy bonito y con una apariencia simple pero eficiente, en cuanto a características y tecnología, va siempre muy por detrás del resto de navegadores.



# Páginas web



## • Los navegadores web

- Hay muchos navegadores disponibles en Internet:

- **Opera:** Siempre ha sido un navegador innovador aunque olvidado. De hecho, fue uno de los primeros que implementó un sistema de compresión de datos para ahorrar tráfico y consumo en dispositivos móviles, tecnología que posteriormente Google Chrome también añadía en su navegador.
- **Tor:** Es un navegador basado en Firefox que integra un sistema que permite la navegación utilizando la red Tor, la cual permite salvaguardar la privacidad del usuario y que cuando navegamos por una página, seamos un poco más anónimos de lo que solemos ser con el resto de navegadores.
- Será por navegadores.... ¿Cómo elijo yo uno? ([enlace](#) y [enlace](#))

Vivaldi



Brave



Blisk



Maxthon



QuteBrowser



**Vídeo:** Guerra de navegadores por uso en PC y móvil desde 1993. Hay que mirar nuestra página web en diferentes navegadores!!

# Páginas web

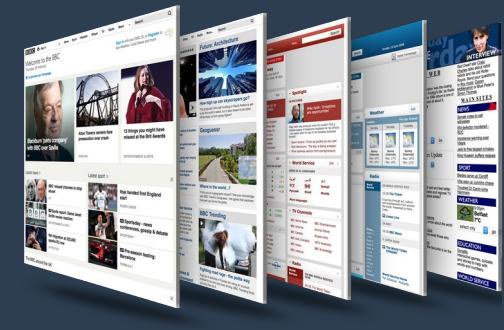


- Pero entonces, ¿Qué es una **página web**?
  - Una página web es un sitio web (website) compuesto por un conjunto de ficheros enlazados entre sí y a las que se accede bajo una dirección **URL**.
- En un servidor, estos ficheros se encuentran desde un directorio raíz (si es local) o externo.
- A partir de una carpeta raíz, donde estará el fichero index.html, se irá pasando a otras de acuerdo con la estructura de directorios como en la imagen.
- El acceso a estos recursos se puede hacer de manera relativa o absoluta.

Nº	RUTA ABSOLUTA	HOME	RUTA RELATIVA
1	var/www/html/lib	Html	Lib
2	tem/etc/bin/mnt	Etc	Bin/mnt
3	bin/etc/mnt/lib	\	Bin/etc/mnt/lib

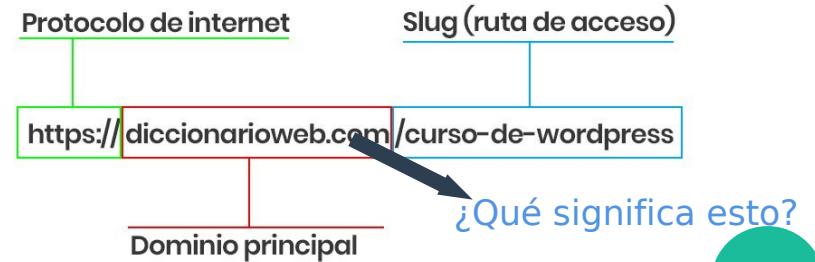


# Páginas web



- **¿Qué es una URL?**
  - Es la abreviatura de Uniform Resource Locator y se define como la dirección de un documento, página web y cualquier otro recurso en Internet. La URL es una secuencia de caracteres y números en ASCII.
- **¿Qué formato tiene?**
  - **Protocolo:** Identifica el protocolo web que se está utilizando y puede ser http://, https://, ftp://, file:// o mailto:..
  - **Dominio principal:** Indica el nombre de la web y puede estar precedido por www. El dominio principal tiene al final una extensión de dominio y también podría tener subdominios
  - **Ruta de acceso (Slug):** Indica la ruta de acceso al recurso en la web dentro del dominio

**Vídeo:** ¿Sabes lo que es una URL amigable? ¿Y una URL estática o dinámica? ¿Sabes qué es un subdominio?



# Páginas web

- Nosotros vamos a trabajar en un servidor local en nuestro ordenador. Pero, ¿qué es un servidor?
  - El servidor es un ordenador en el que se alojan webs y servicios de red para que los usuarios puedan acceder utilizando el protocolo cliente-servidor.
- Claro está, a nuestro ordenador en una red privada no van a acceder desde fuera, necesitamos darle un nombre público disponible en la red. Para esto se creó el DNS.

**Vídeo:** ¿Montamos nuestro servidor local para que accedan desde fuera o usamos un VPS?

- Una medida rápida para empezar es usar empresas que se dedican al alojamiento web y la reserva de nombres de dominio. Algunas de las empresas de hosting más populares están en el siguiente [enlace](#)



Ejemplo IONOS (análisis)



# Páginas web

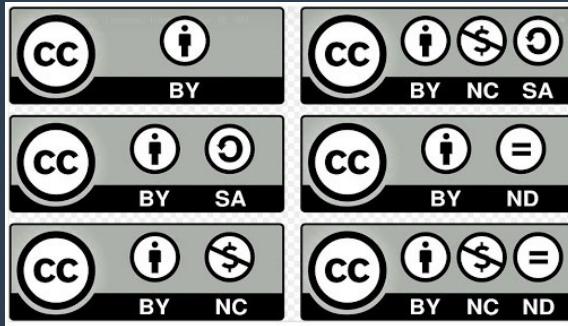
- **¿Pero si todavía no hemos diseñado nada y ya estamos pensando en hacer hosting?**
- A lo largo de este tema aprenderás HTML pero en su desarrollo profesional tienes que considerar los siguientes aspectos:
  - **Objetivo:** ¿Qué se pretende conseguir con la web?
  - **Público:** ¿A quién se dirige? ¿Puede usarla alguien con discapacidad?
  - **Contenido:** ¿Qué contenido tiene? ¿Formato: texto, imágenes, tablas, multimedia, animaciones, hipervínculos
  - **Directrices:** ¿Qué especificaciones se usarán en cuanto a forma, organización y estructura?
  - **Usabilidad:** ¿Se qué hay que hacer en cada momento? ¿Encuentro rápido las cosas?
  - **Tecnología:** ¿Con qué medios técnicos se cuenta para realizarla?
  - **Tiempo:** ¿De cuánto tiempo se cuenta para realizarla?
  - **Presupuesto:** ¿Cuánto cuesta el alojamiento, el servidor o el desarrollo?

# Páginas web

- Aunque no es el tema de esta materia, también hay que considerar aspectos visuales en su diseño como:
  - Contraste del texto y el fondo
  - Tamaño de las imágenes o texto
  - Elección de los colores
  - La interfaz de acceso: PC, tablet o móvil (Esto si lo trataremos con **RWD**)
  - La organización de los objetos en la web
  - Comodidad para la lectura en la pantalla (letras sin adornos)
- Aquí hay unos ejemplos de mal diseño: ([enlace](#))

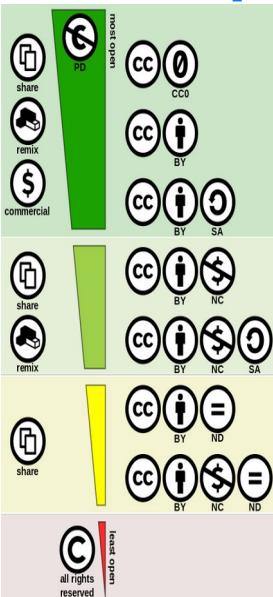


# Páginas web



- **¡OJO CON LAS LICENCIAS EN LOS CONTENIDOS USADOS!**

Cualquier editor de un sitio web debe conocer las licencias aplicables a los contenidos digitales que genera o que utiliza de terceros. Las licencias más habituales son:



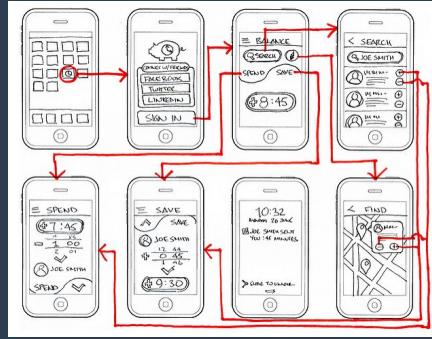
- **Copyright (c).** Esto indica que la obra o recurso con esta licencia establecida no se debería emplear, si no es con el consentimiento expreso del autor.
- **Creative Commons (CC), GPL.** Licencia de tipo abierto generalmente el material se puede emplear, incorporar, modificar y difundir, pero siempre debemos ceñirnos a lo que se indique en la licencia: si se nos indica aspectos como que debemos citar la autoría o si no podemos hacer modificaciones. No requiere pedir consentimiento, sólo seguir las pautas indicadas.
- **Dominio Público (CC0).** Las obras que están en el dominio público pueden ser utilizadas modificadas o adaptadas libremente.
- Si en el material no hay nada indicado o no es posible localizar su fuente, sería mejor preguntar y no emplearlo, ya que podríamos encontrarnos con que su autor nos reclame su reconocimiento de autoría en un futuro.
- Hay bancos de imágenes libres [aquí](#) o de vídeos [aquí](#).

**flickr**



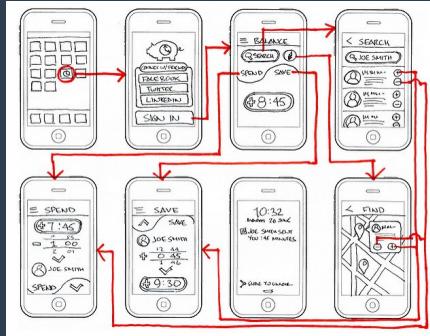
**vimeo**

# Prototipado web

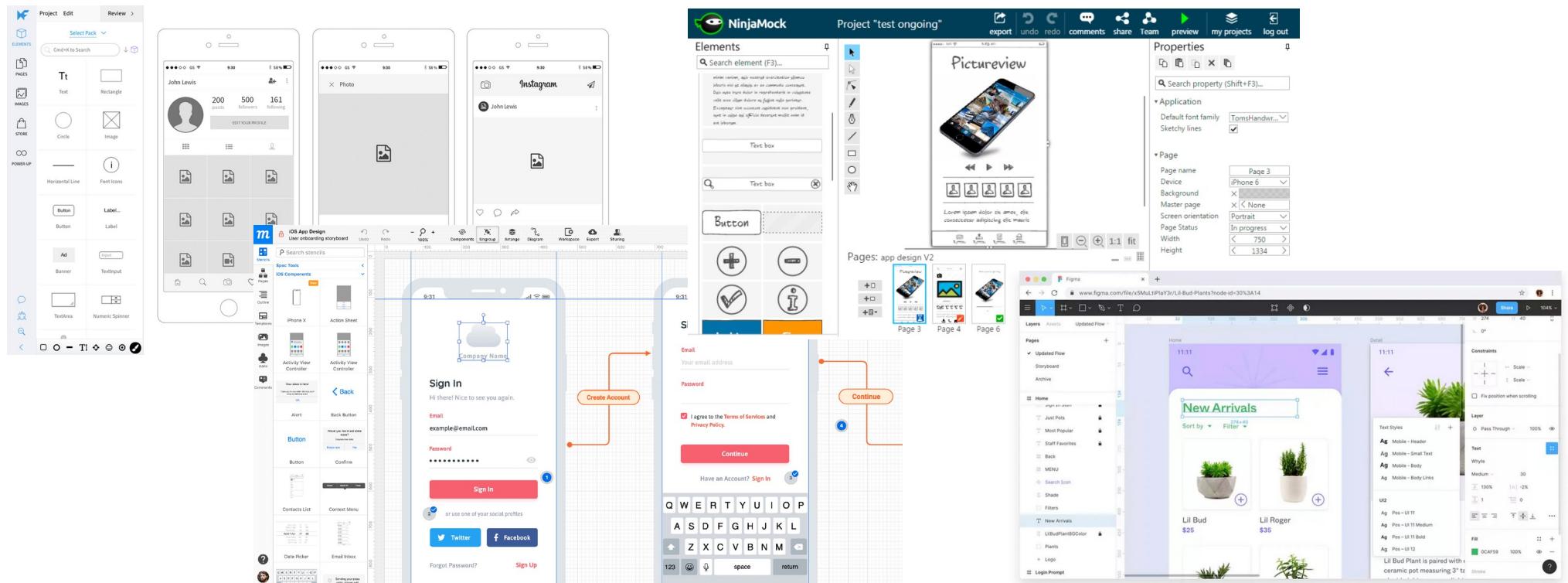


- Antes de escribir una sola línea de código, tenemos que conocer muy bien el diseño de la aplicación, sus funciones y el uso que le va a dar el cliente.
- Aunque esto se debe obtener durante la entrevista, a menudo el cliente NO es técnico, por lo que es conveniente mostrarle un prototipo o **wireframing**.
- ¿Qué ventajas tiene el wireframing?
  - **Rápidos y baratos de crear:** Como son bocetos esquemáticos son rápidos de crear y tienen un coste muy bajo. Esto te permite realizar múltiples versiones hasta encontrar la adecuada sin que ello suponga un problema de tiempo o dinero.
  - **Detectar y corregir los problemas antes:** Al ser sencillos y rápidos de realizar te permiten exponerlos rápidamente a feedback y resolver problemas básicos relacionados con la usabilidad y funcionalidades propuestas.
  - **Mejoras sencillas:** En poco tiempo podrás mostrar los primeros Wireframes a amigos, clientes o repasarlos tu mismo para repasar las mejoras que se puedan realizar en el diseño, el posicionamiento de los elementos o la estructura de los contenidos.
  - **Mejor usabilidad:** Planear previamente la estructura y los elementos de la página web te permitirá ofrecer una mejor usabilidad al no improvisar sobre la marcha y haber definido previamente estos elementos.

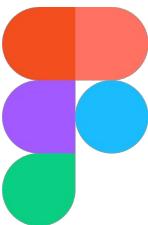
# Prototipado web



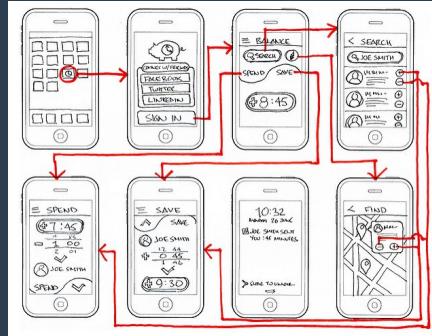
- Algunas herramientas de prototipado puedes ser para web o escritorio y estos son ejemplos de las interfaces que se usan:



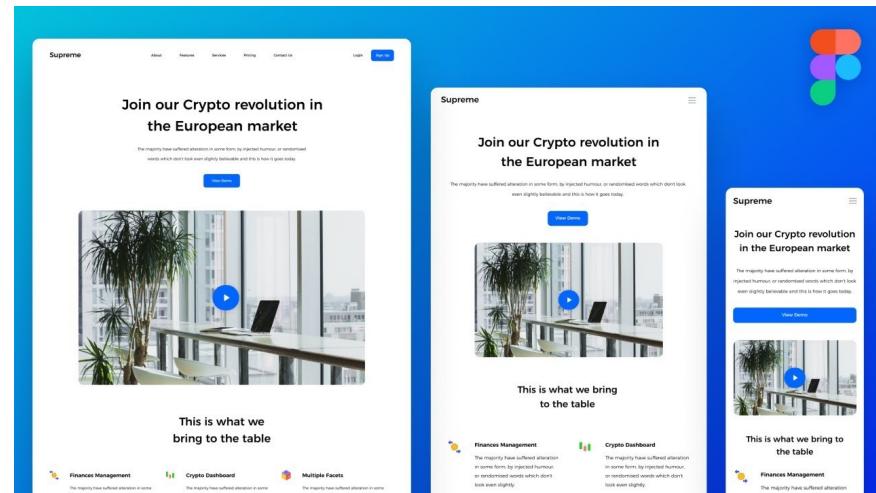
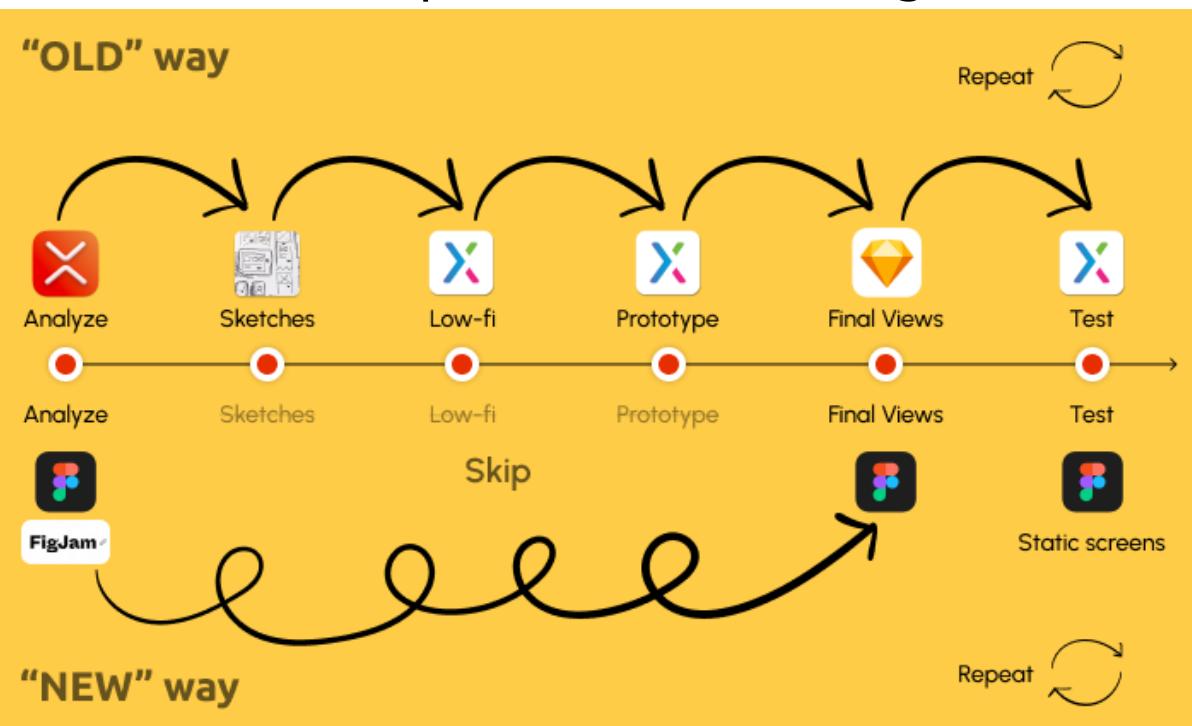
- Vamos a ver cómo funciona Figma en [OpenWebinars \(TAREA\)](#)
- No es el único, otros son: mockflow, wireframe, ninjamock



# Prototipado web

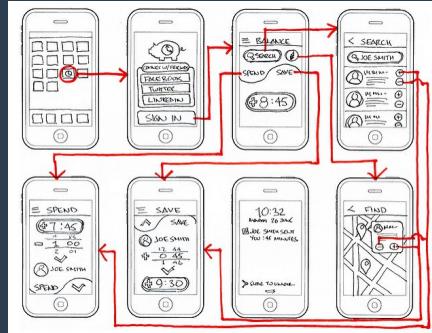


- Herramientas intuitivas como Figma permiten acelerar el proceso de prototipado clásico y que se puedan desarrollar de manera colaborativa y estén disponibles en la nube.
- A veces ya no es sólo tratar el diseño de acuerdo a una plataforma, sino también cómo se van a organizar los componentes al ver la misma web en plataformas diferentes. Esto lo veremos más adelante como Responsive Web Design.

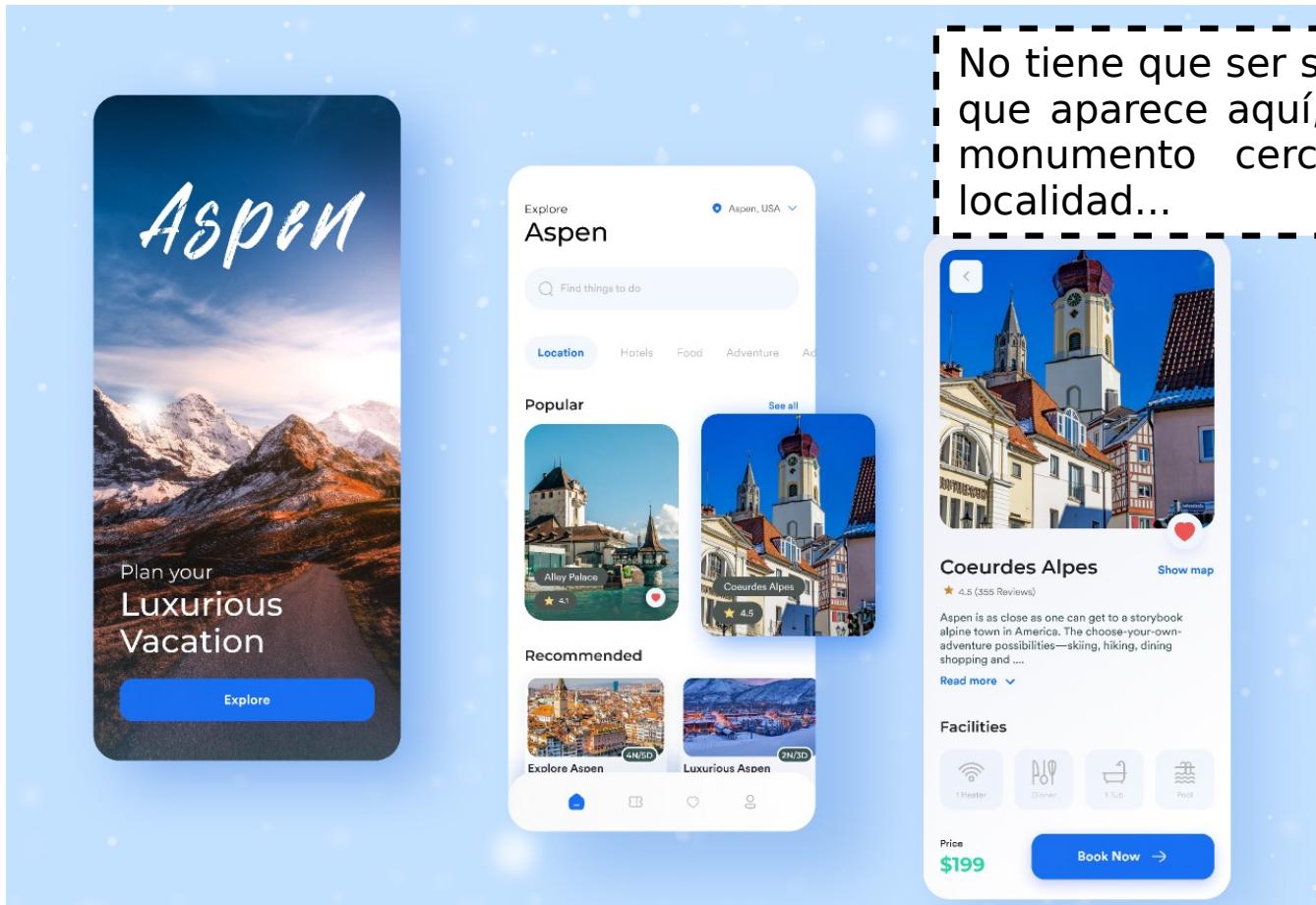


Enlace Youtube

# Prototipado web



- **Actividad:** Desarrolla un prototipo en FIGMA para móvil similar al de la imagen. **NO** tiene que ser funcional el prototipo.



No tiene que ser sobre la misma información que aparece aquí, puedes poner otra de un monumento cercano, parque natural, tu localidad...

Cuando entregues la tarea pondrás el enlace como este: ([enlace](#)) y que se obtiene al pinchar en el proyecto → botón derecho → copy Link

# ¿Cómo escribiremos HTML?



- En la práctica, se utilizan **editores e IDEs** para facilitar el diseño de páginas web como son: Notepad++, Brackets, Sublime text, Atom, **Visual Studio Code**, Komodo Edit o Netbeans



- [Aquí](#) puedes encontrar información de los más destacados y el debate de editor vs IDE, donde para desarrolladores profesionales se usa VSCode, Netbeans o Komodo. Aquí verás más información sobre IDEs utilizados en desarrollo web.
- Por supuesto hay otros editores tipo **WYSIWYG** como: TinyMCE, BlueGriffon, HTML notepad, CKEditor, Dreamweaver o blogspot. Están orientados a principiantes.

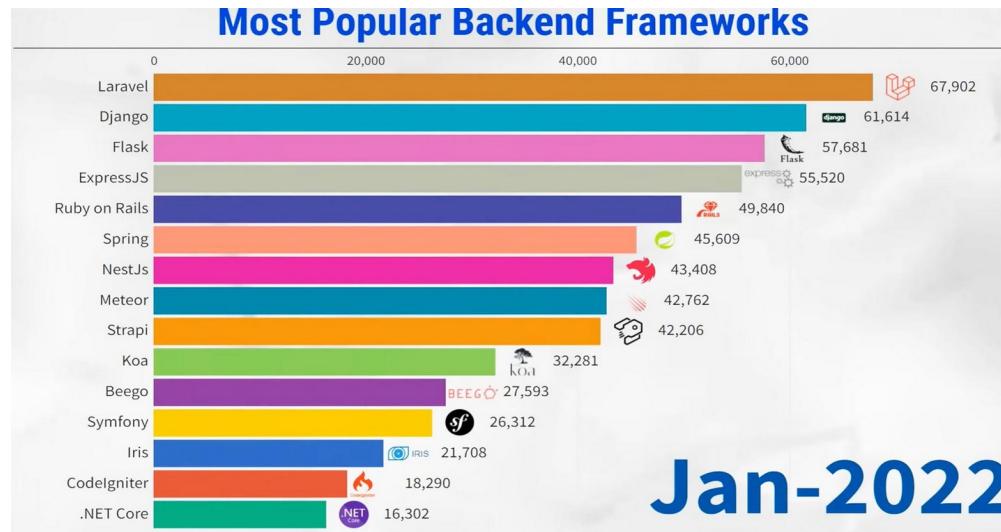


- Ni que decir tiene que poco a poco van ganando terreno los **frameworks** que facilitan el desarrollo con un sinfín de facilidades para centrarse en la lógica del funcionamiento de un sitio web (sobre todo backend), en lugar de gastar el tiempo en la escritura de códigos que se pueden reutilizar. [Aquí](#) puedes encontrar los 9 más destacados como: Angular, Laravel o Symfony

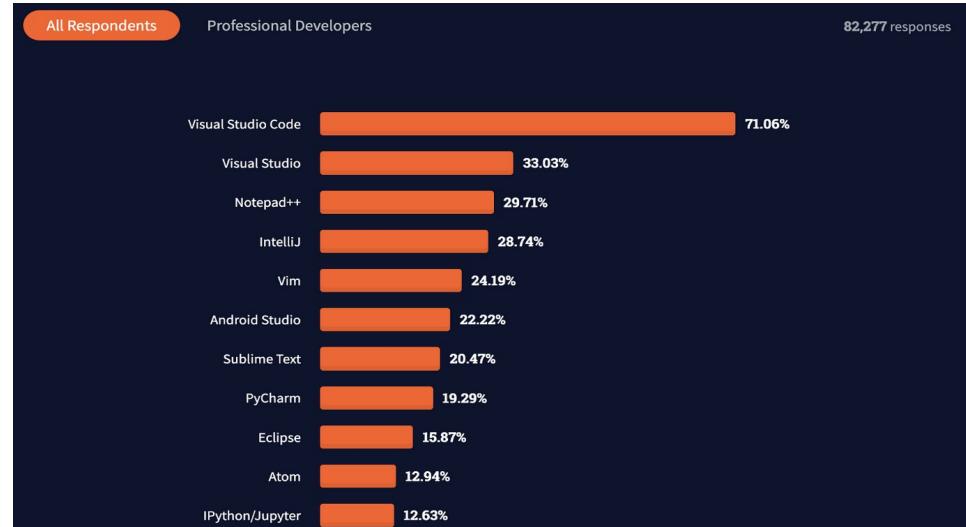
# ¿Cómo escribiremos HTML?



- El uso de los frameworks se ha popularizado mucho en los últimos años:



Jan-2022



- Utilizar un framework supondría tener ya la base de desarrollo web. Nosotros utilizaremos VSCode en principio para el desarrollo de HTML, CSS y JS desde cero. Para facilitar el desarrollo haremos uso de las extensiones que se indican a continuación: Live Server, Tag rename, Git Lens, Emmet (integrado)



- **Actividad:** Antes de empezar a escribir código vamos a hacer uso de un editor WYSIWYG para crear nuestro blog personal con dos entradas. Poned la dirección del blog o una captura del mismo.



# Control de versiones GIT



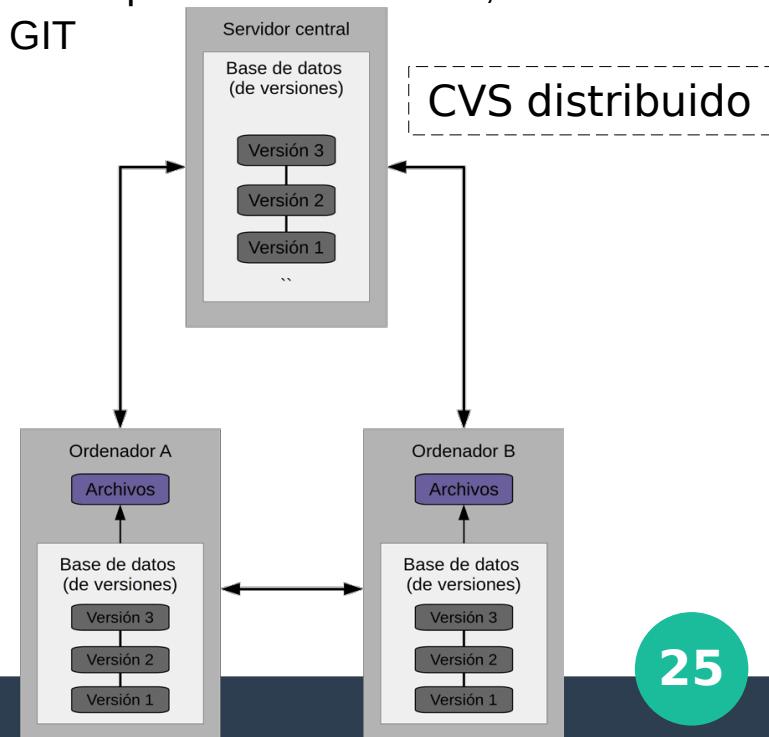
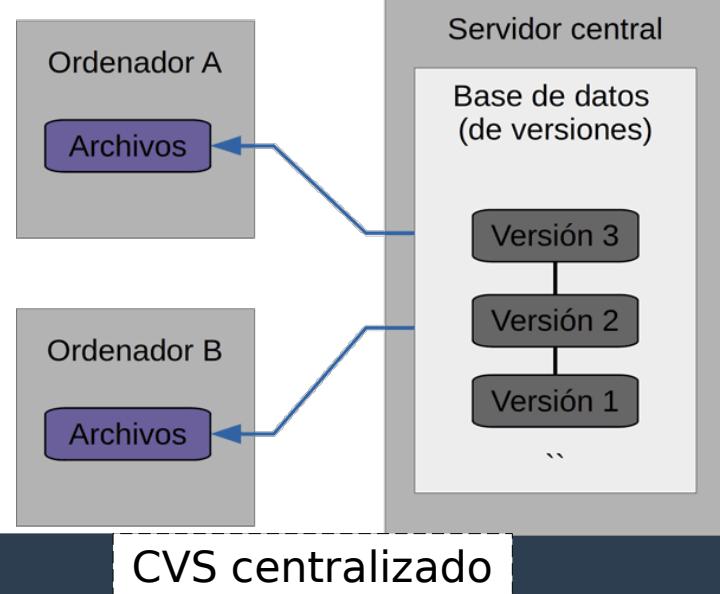
- Los desarrolladores de código necesitan herramientas que le garanticen tener acceso a la información en cualquier lugar, que puedan participar varios usuarios sobre ellas y que permitan control de versiones por si hay errores o queremos ir a una versión anterior.
- Tradicionalmente esto se ha hecho en portales donde colocábamos ficheros tipo documento\_v1, documento\_v2,... Al final generamos mucha cantidad de ficheros que luego hay que ir borrando y tienes que acordarte por qué creaste la v2 frente a la v1. ¿No hay alguna herramienta que gestione esto más eficiente?
- Un sistema de control de versiones (VCS en inglés) no es más que un software capaz de registrar los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.
- Hoy se usan en multitud de proyectos, pero su auge surge a raíz del desarrollo del Kernel de Linux
- En el mercado podemos encontrar muchos [CVS](#):



# Control de versiones GIT



- Hay 3 tipos de CVS:
  - **Local:** Este tipo de control de versiones, permite gestionar un proyecto en tu computadora de forma local. Un sistema de este tipo, es útil cuando trabajas en un proyecto independiente. No es muy práctico
  - **Centralizado:** Estos sistemas tienen un único servidor que contiene todos los archivos versionados y varios clientes que descargan los archivos necesarios desde ese lugar central. Este fue el estándar para el control de versiones por muchos años y aún se utiliza bastante.
  - **Distribuido:** Ya no es necesario estar conectado constantemente al servidor, ya que es posible realizar cambios y almacenarlos en la copia local para luego, cuando sea posible o necesario, actualizar los datos al servidor. Es el que más se utiliza en la actualidad con GIT
- Veamos la diferencia de estos dos últimos modelos:



# Control de versiones GIT



- Hablemos de **GIT**:
  - Es un software de control de versiones distribuido desarrollado por Linus Torvalds, para optimizar el trabajo en proyectos que cuenten con un gran número de archivos.
  - Es un proyecto de código abierto multiplataforma que se inició en 2005 y es el más popular en el mercado, ya que lo usan el 87% de los desarrolladores.
  - Existen un montón de plataformas de desarrollo colaborativo (también conocidas como forjas) que admiten el uso de Git como sistema de control de versiones. Estas son: GitLab, Bitbucket, Codegiant, SourceForge o GitHub
- Hablemos de **GitHub**:
  - Es un servicio basado en la nube que aloja el CVS GIT. Éste permite a los desarrolladores colaborar y realizar cambios en proyectos compartidos, a la vez que mantienen un seguimiento detallado de su progreso.
  - Su popularidad radica en que hace accesible el CVS a personas con pocos o ningún conocimiento técnico. Permite ver ejemplos de código, participar, tener una copia (fork),...
  - Además, se puede utilizar como portfolio y una forma de demostrar la experiencia profesional en diferentes proyectos de cara a conseguir un trabajo.

# Control de versiones GIT



- **Práctica:** Vamos a seguir los siguientes pasos para tener nuestro propio GitHub y preparar el entorno de trabajo que usaremos en prácticas.

- Pasos a seguir:

- [Instalación de GIT \(LINUX\)](#), [Crear una cuenta GitHub](#)
- [Funcionamiento de GIT con ejemplos](#)
- [Crear un repositorio en GitHub y publicar](#)
- [Enlazar GIT desde VSCode](#)

Prueba a documentar con [markdown](#) el fichero Readme.md

**NOTA:** Hemos visto la base de GIT y su manejo con VSCode pero puedes ver más detalle de esto con el documento de Moodle de prácticas, en este [enlace](#) o [Youtube](#)

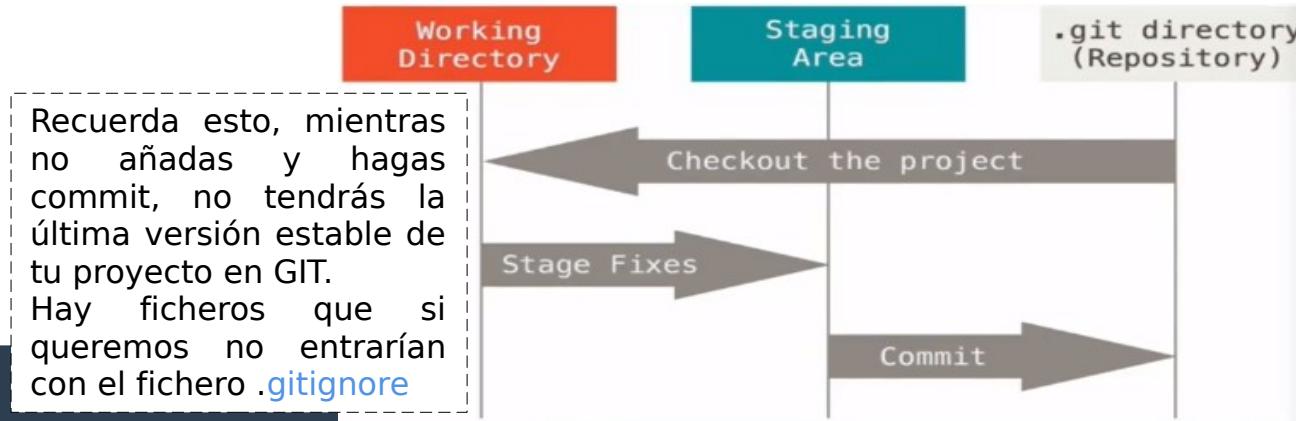
Si quisieramos hacer lo mismo pero con SVN podríamos seguir el siguiente tutorial ([enlace](#))

En Moodle tenéis una chuleta con los comandos de uso más importantes de GIT y otros tutoriales

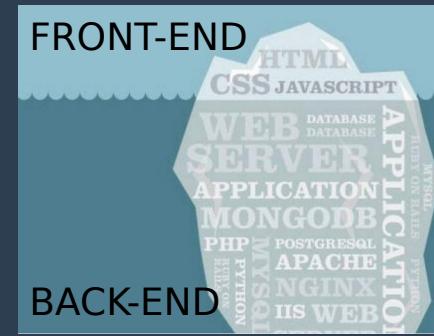
**Tarea:** Obtén el certificado **GIT** de OpenWebinars

**Importante**, nunca dejes tus credenciales activas en git config si hay más de un usuario que usa tu ordenador. Mira [git config unset](#) o desde [Windows](#)

Recuerda esto, mientras no añadas y hagas commit, no tendrás la última versión estable de tu proyecto en GIT. Hay ficheros que si queremos no entrarían con el fichero `.gitignore`



# ¿Cómo funciona la web?



- Cuando un usuario realiza la petición de una página web hay que tener presente que se producen dos procesos. El primero es el que solicita los recursos para formar la página web a mostrar y que conlleva procesos como: interpretar código del lado del servidor (por ejemplo en PHP o ASP.Net), pedir recursos a otros servidores (bases de datos, mapas, streaming de vídeo, etc.) o almacenar datos de sesión. El segundo es el encargado de darle forma a ese contenido para que el navegador lo interprete. Así tenemos el desarrollo **front-end**, **back-end** y **full stack**.

**Vídeo:** ¿Qué es cada uno y qué solicitan las empresas?

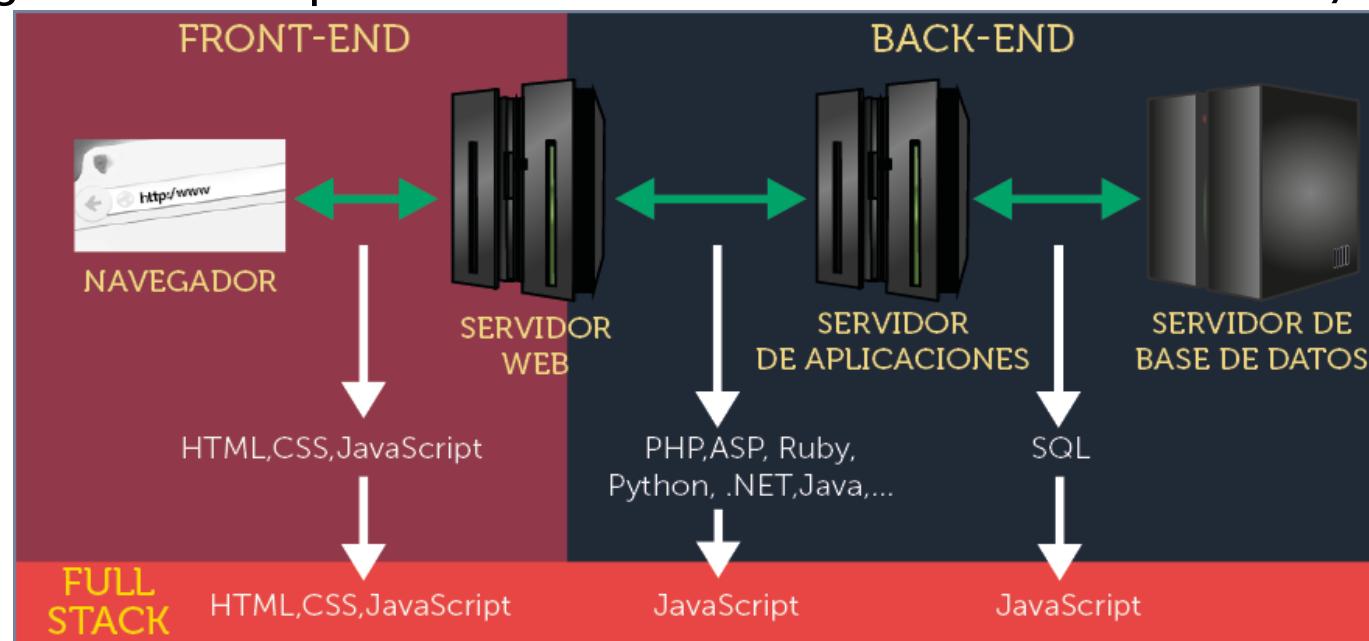
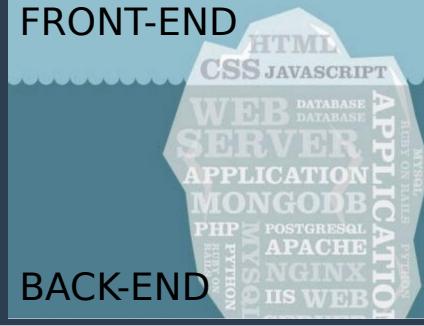


Ilustración 2-3. Esquema completo de tecnologías habituales en front y back-end. Se destaca la técnica Full Stack que usa JavaScript en todas las capas.

# ¿Cómo funciona la web?

FRONT-END

BACK-END



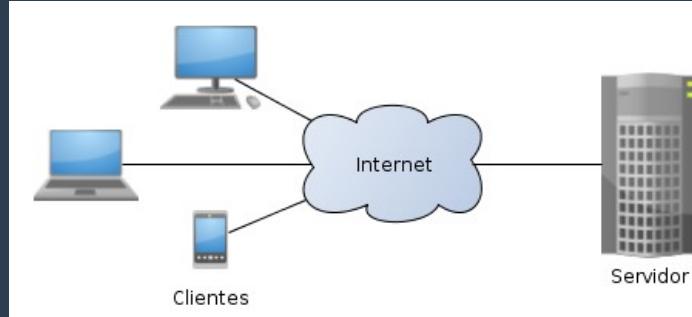
- Vamos a ver qué significa estos enfoques de desarrollo, ya que a menudo son equipos diferentes en una empresa grande. El problema es que si te dan un proyecto completo tendrás que ser Full Stack que sabe de todo. Es decir, puedes tener tendencia hacia una de las ramas pero debes dominar ambas.
  - **Front-End** es la parte de una aplicación que interactúa con los usuarios, es conocida como el lado del cliente. Básicamente es todo lo que vemos en la pantalla cuando accedemos a un sitio web o aplicación: tipos de letra, colores, adaptación para distintas pantallas(RWD), los efectos del ratón, teclado, movimientos, desplazamientos, efectos visuales... y otros elementos que permiten navegar dentro de una página web. Un desarrollador front end debe conocer los siguientes lenguajes de programación: HTML5, CSS3, JavaScript, Jquery, Ajax.



- **Back-End** es la parte centrada en el servidor donde se toman los datos, se procesa la información y se envía al usuario en formato HTML. Esta parte es bastante más densa y por ello, un desarrollador Back end debe tener amplios conocimientos de los siguientes lenguajes: frameworks y los tipos de base de datos. No siendo necesario conocer todos los lenguajes pero sí entender y saber trabajar con algunos de ellos o con frameworks que facilitan esta tarea como React o Angular.



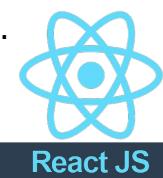
# ¿Cómo funciona la web?



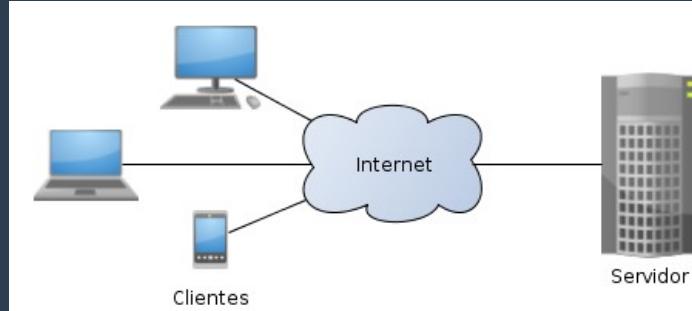
- **Modelo cliente-servidor:**

- En la WWW se sigue la arquitectura cliente-servidor, que determina en qué lado se ejecutan, en cliente o en servidor, los lenguajes de programación que permiten generar la estructura de la página web.
- En el lado del servidor:
  - **Java, PHP, .NET, Ruby, Python:** estos lenguajes procesan una petición de un usuario mediante la interpretación de un script (código) en el servidor web para generar páginas HTML dinámicamente como respuesta.
  - **SQL:** lenguaje de acceso a base de datos que permite la consulta , las inserciones o modificaciones sobre la misma.
- En el lado del cliente:
  - **HTML:** lenguaje de marcas de hipertexto esta formado por etiquetas que describen y estructuran el contenido de una página Web. Se considera esencial para el desarrollo de la WWW cuyo desarrollo regula la W3C.
  - **CSS:** hojas de estilo en cascada es un lenguaje desarrollado para definir y crear la presentación (estilo) de un documento estructurado escrito en HTML.
  - **JavaScript:** lenguaje de programación interpretado por el navegador o cliente web. Este lenguaje permite añadir dinamismo a la página modificando su estructura y estilo entre otras cosas. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web, evitando realizar peticiones al servidor.
- En este curso nos centraremos en los lenguajes del lado del cliente que permiten desarrollar la interfaz gráfica del usuario, en otras palabras, el front-end. No obstante, también haremos desarrollo de API con React.

Vídeo: ¿Cómo funciona REACT y por qué es tan popular?



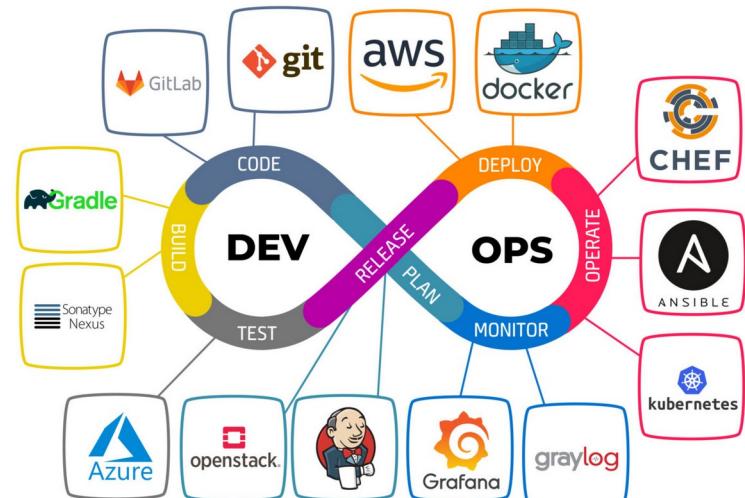
# ¿Cómo funciona la web?



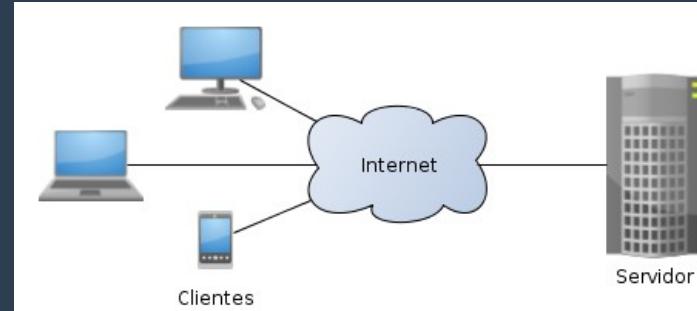
- **Modelo cliente-servidor:**

- Normalmente no hay separación estricta entre el mundo Front-end y Back-end. No hace falta que seas Full-stack pero sí es necesario que ambos sistemas mantengan comunicación para que no haya errores en el proceso de diseño e implantación de una aplicación web. A esto lo llamaremos **DevOps** y esto es una tendencia muy demandada en las empresas.
- DevOps es una metodología de desarrollo de aplicaciones de calidad, donde para establecer mejor comunicación se realiza una coordinación a través de herramientas comunes. TODOS participan durante el desarrollo y con seguridad se incluye DevSecOps.
- ¡Ojo! Esto requiere experiencia y tiempo
- Otra metodología sería SCRUM

**Vídeo:** ¿Cómo funciona DevOps?

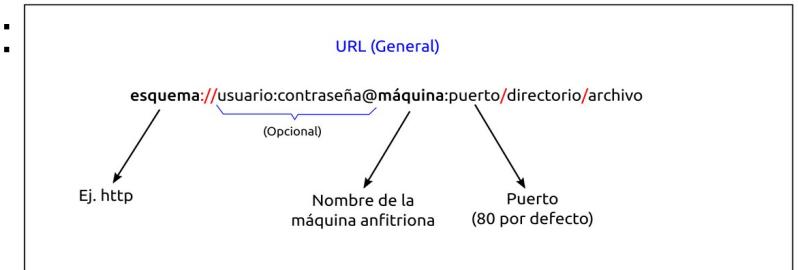


# ¿Cómo funciona la web?

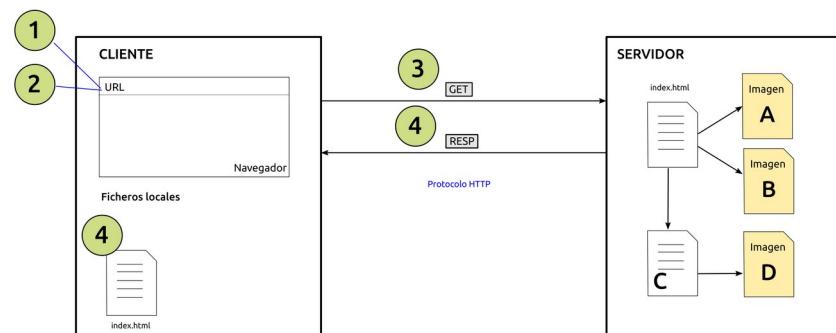


- **Acceso a una web:** Para acceder a un recurso en Internet tenemos que conocer su URL que tiene el siguiente formato:

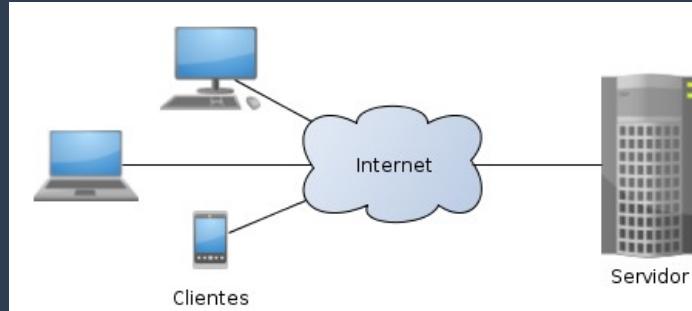
Tened en cuenta que al comunicarnos también mandamos información de nuestro sistema y localización. Esto lo utilizará el servidor para mandarnos la información en nuestro idioma



- **¿Qué es lo que ocurre desde que se introduce la URL en el navegador?**
  - Fase 1: El usuario introduce la URL en el navegador. Esta URL puede estar abreviada: www.urjc.es
  - Fase 2: El navegador convierte la URL en el formato correcto: http://www.urjc.es:80/
  - Fase 3: El navegador solicita el documento raíz (index.html) al servidor, a través de una solicitud GET del protocolo HTTP
  - Fase 4: El Servidor devuelve el fichero index.html tras acceder a los recursos necesarios (ficheros, bases de datos, URL,...). Queda almacenado en la máquina del cliente en su caché.



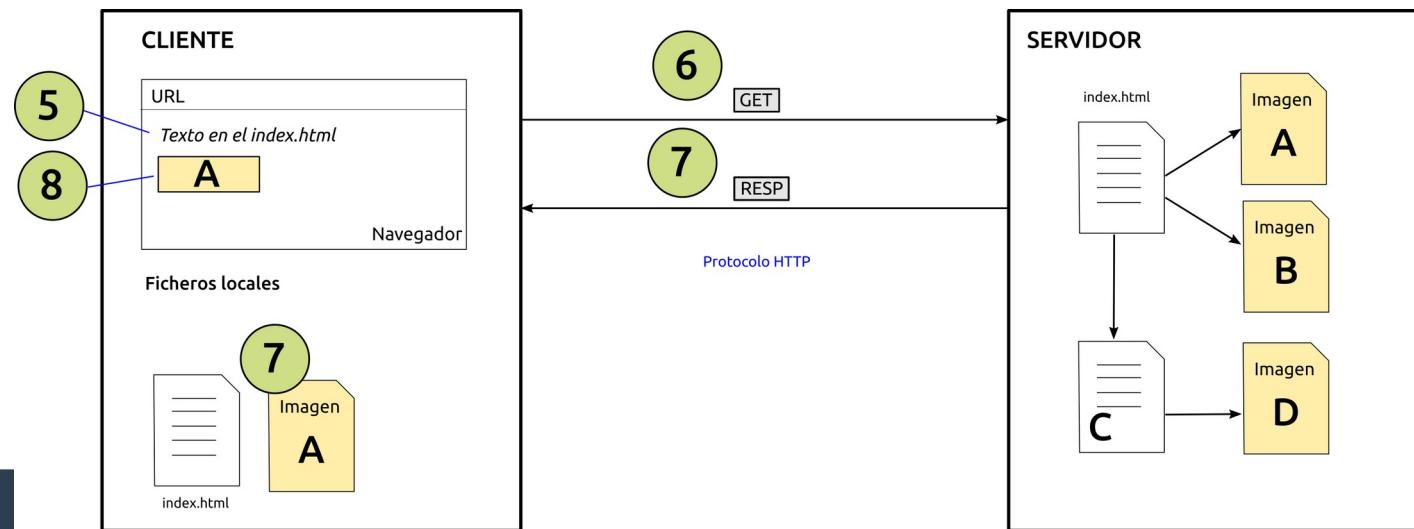
# ¿Cómo funciona la web?



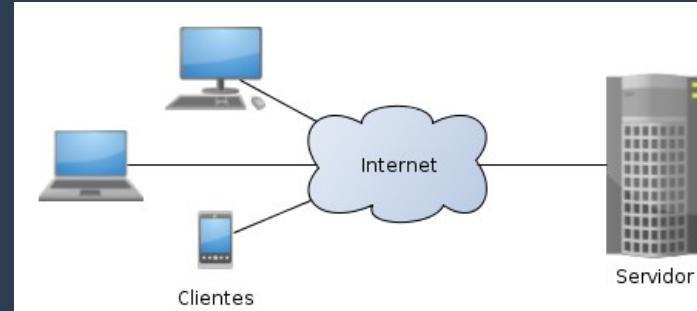
- **¿Qué es lo que ocurre desde que se introduce la URL en el navegador? (cont.)**

- Fase 5: El navegador abre el fichero index.html y lo analiza. Muestra al usuario su contenido
- Fase 6: El navegador detecta que hay una imagen (A). Pide al servidor el fichero con esa imagen
- Fase 7: El servidor devuelve la imagen A y se almacena en el cliente
- Fase 8: El navegador muestra la imagen

Se irán repitiendo los procesos 6 a 8 hasta que se carga toda la página completa en el navegador de acuerdo con su estructura DOM y después su funcionalidad (JavaScript) y el resultado visual (CSS)



# ¿Cómo funciona la web?



- **¿Qué es lo que ocurre desde que se introduce la URL en el navegador? (cont.)**
- Durante esta comunicación se producen una serie de códigos de estado de respuesta HTTP:
  - Respuestas informativas (100–199),
  - Respuestas satisfactorias (200–299),
  - Redirecciones (300–399),
  - Errores de los clientes (400–499),
  - Errores de los servidores (500–599).
- Como usuarios de Internet con el navegador, estamos del lado del cliente, por eso nos suena más el error de cliente (el más conocido 404) y que el programador puede hasta dar formato.

| Veamos que significan estos códigos [aquí](#) |

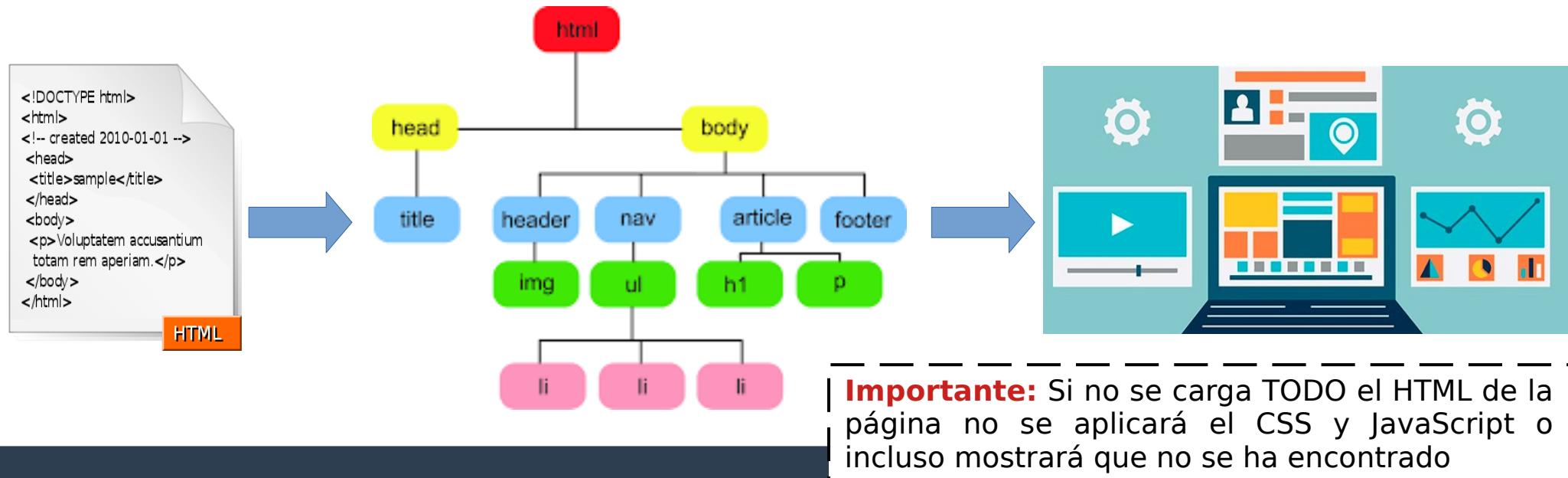
| Cuando hagas un proyecto web tienes que asegurarte que todos los enlaces funcionan y generar la página 404 corporativa ([consejos](#)) |





# Estructura del HTML

- Al igual que XML, los documentos HTML se encuentran estrictamente organizados por etiquetas y contenido. De manera que hay que cumplir las mismas reglas que con XML: estructura declarativa y marcado riguroso
- La diferencia con XML es que cada parte del documento está diferenciada, declarada y determinada por etiquetas específicas que el navegador interpreta y muestra al usuario.
- Así, a partir de un fichero .html tras interpretar su contenido, el navegador va a generar el árbol DOM y luego mostrar al usuario.



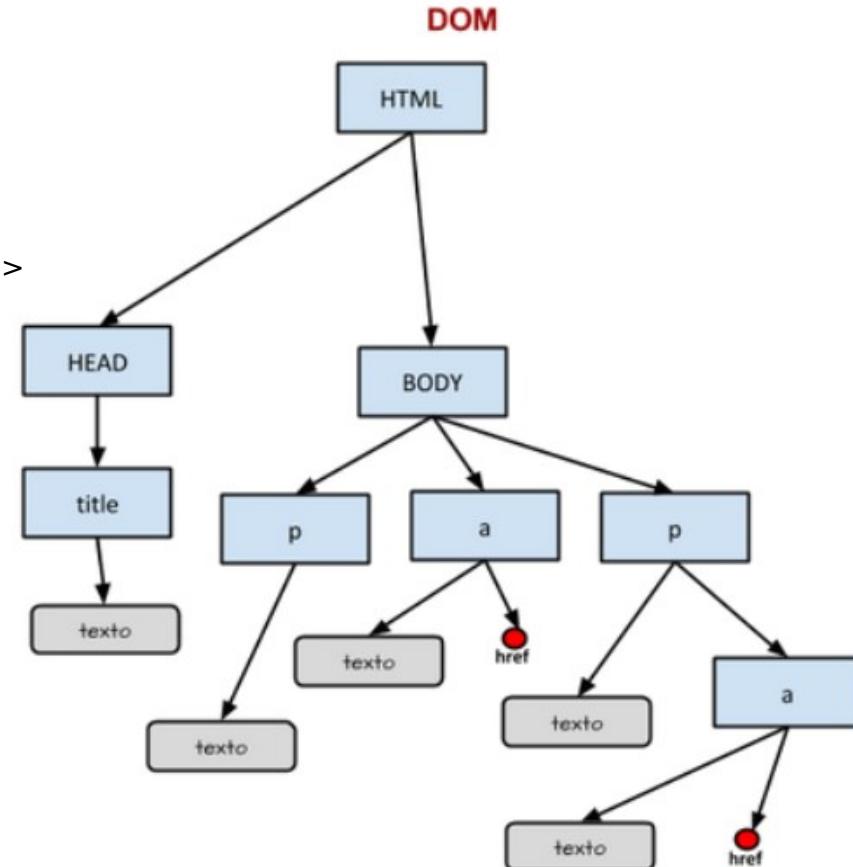
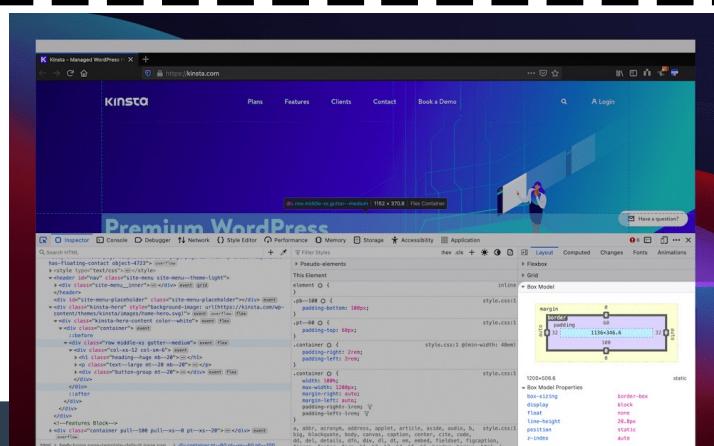


# Estructura del HTML

- Ejemplo de código HTML y su árbol DOM correspondiente:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo de elementos en línea y elementos de bloque</title>
  </head>
  <body>
    <p>Los párrafos son elementos de bloque.</p>
    <a href="http://www.google.com">Los enlaces son elementos en línea</a>
    <p>Dentro de un párrafo, <a href="http://www.google.com">los enlaces</a>
      siguen siendo elementos en línea.</p>
  </body>
</html>
```

Si inspeccionamos el resultado en el navegador podremos ir viendo un modelo de cajas para montar la página web y a medida que recorres el árbol vas viendo su contenido. Hablaremos de estos luego en CSS. Este modo de inspección de los navegadores puede ser muy útil para probar configuraciones en lugar de ir cambiando y probando cada aspecto.



Antes de desarrollar nada, prepara bien el entorno de trabajo y recuerda hacer copias en el repositorio GIT

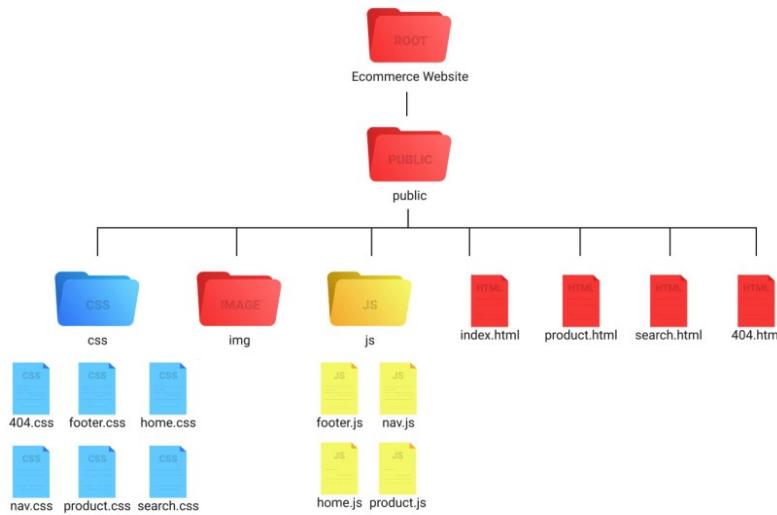
# Estructura del HTML



- Antes de empezar a tratar el contenido de HTML, hay que intentar trabajar de manera ordenada. Esto no quiere decir que el código que se genere en la aplicación no sea complejo, sino que haya una metodología de trabajo que ayude al desarrollo. Hay hasta un principio de diseño: KISS



- Hay que pensar que un diseño ordenado y estructurado nos va a facilitar la tarea de desarrollo y su posterior mantenimiento.
- Por este motivo, siempre que creamos un proyecto web, o de programación, es conveniente que creamos directorios donde diferenciamos partes del proyecto por funcionalidad (secciones de la web) o por contenido (CSS, JS, imágenes, documentación).





# Estructura del HTML

- El problema de hacer HTML es que tenemos que hacer documentos correctos y válidos como ocurre en XML, de lo contrario, podemos tener problemas con la generación de ese árbol DOM y su posterior visualización.
- **Práctica:** Vamos a ver lo que se aprecia con el siguiente ejemplo y desde VSCode trataremos de corregir los errores. De paso estableceremos el entorno de trabajo

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
"http://www.w3.org/TR/html4/strict.dtd">  
  
<head>  
    <title>Título del documento</title>  
</head>  
<body>  
    <H1>Lenguajes de Marcas y Sistemas de Gestión de la Información</h1> <br>  
    <h2>Unidad 1: Introducción a los lenguajes de marcas  
    <p>párrafo en el cuerpo <br> del documento.</p>  
    <hr>  
</body>  
</html>
```

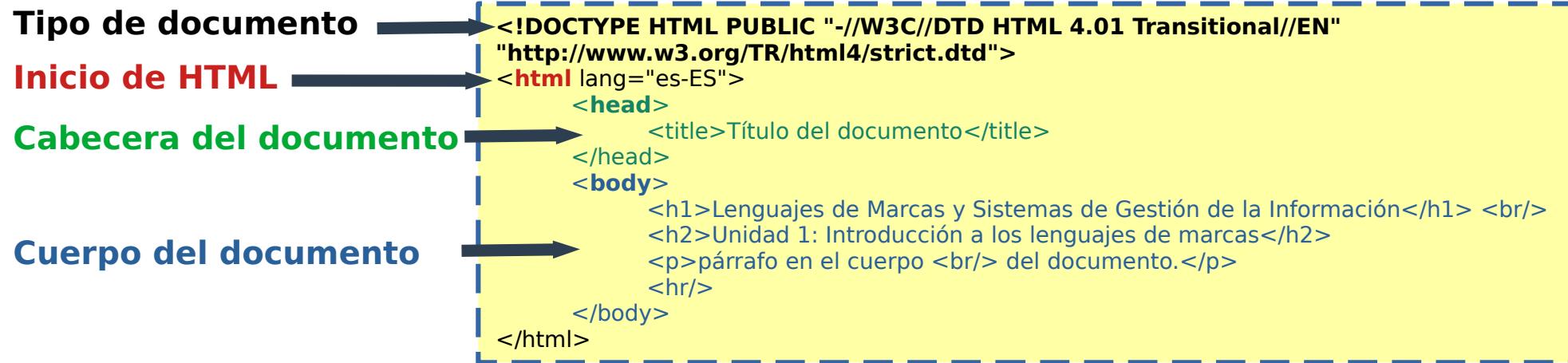
- Fijaros que a pesar de que hay errores, el navegador “intenta” mostrar una página web. Sólo si el error es grave nos daríamos cuenta de ello en códigos muy grandes.
- Por este motivo, es necesario que el HTML sea **correcto** y **validado**. Para ello, podemos optar por páginas web como **W3C** o renombrando el fichero a XHTML o ya desde los propios IDE o frameworks como pasa con VSCode ([enlace](#))

En teoría, todas las páginas deberían estar validadas y poseer algún icono como el de la imagen. ¿Qué significa **AAA**? Ejemplo que sí pasa el test ([enlace](#))



# Estructura del HTML

- Partiendo del ejemplo propuesto y corregido, podemos distinguir varias partes de un documento HTML:



- En cuanto al **DOCTYPE**, se encuentra al comienzo del fichero y es equivalente a un DTD externo público de XML donde se comprueba si la sintaxis es correcta. Hasta la llegada de HTML5 teníamos que decidir el tipo de comprobación como: HTML 4.01 Strict (Estricto), HTML 4.01 Transitional (Transicional), XHTML 1.0 Strict, y varios más. Ej: `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">`
- Ahora con HTML5 sólo es necesario poner: `<!DOCTYPE html>`



# Estructura del HTML

- Luego de declarar el tipo de documento, debemos comenzar a construir la estructura HTML con `<html></html>`
- Dentro de las **etiquetas** van a aparecer **atributos**, muchos de los cuales son particulares como alt, border, href y otros globales como:

En el caso de `<html>` tiene atributos particulares y opcionales como son: manifest, version y xmlns

El atributo más utilizado con esta etiqueta es el idioma como `<html lang="es">` por accesibilidad.

Dentro de un mismo idioma, se puede escoger por la zona. ([enlace](#))

Aunque no debe usarse por estar CSS, en los ejemplos lo usaremos para “marcar” zonas y que se vean mejor los conceptos

Atributo	Descripción
class	Permite agrupar elementos. El valor de este atributo es una cadena de caracteres definida por el autor de la página. Si varios elementos comparten el mismo valor, se dice que son de la misma clase. Se utiliza para asignar estilos y durante la programación con JavaScript.
contenteditable	Si existe, el contenido del elemento es editable.
hidden	Si existe, el elemento no se representa en el navegador.
id	Identifica al elemento de manera única en el documento. El valor es una cadena de caracteres definida por el autor de la página. Se utiliza para asignar estilos y durante la programación con JavaScript.
lang	Determina el idioma en el que está escrito el contenido del elemento.
spellcheck	Determina si el elemento debe ser analizado ortográfica y gramaticalmente por el navegador.
style	Permite asignar un estilo al componente.
tabindex	Determina el orden de selección de elementos cuando se pulsa el tabulador.
title	Permite especificar información extra al elemento.
translate	Determina si el elemento debe ser traducido o no.

- Tenéis todo el listado de atributos de HTML5 para cada etiqueta aquí.



# Estructura del HTML

- **Elemento HEAD:**
  - Es la primera sección en el html y aunque no es visible, permite incluir el título, los enlaces a CSS y JS, icono de la web o las keywords.
- Dentro de la cabecera <head> </head> podemos tener los siguientes elementos en cualquier orden: (Mención especial **link**, script y meta)

Elemento	Versión	Descripción
<base></base>	HTML 2.0	URL a partir de las cuales se construyen las referencias href de la página y otros enlaces.
<link></link>	HTML 2.0	Incluye otros documentos enlazados, tales como hojas de estilo o scripts.
<script></script>	HTML 3.2	Especifica un elemento de tipo script, para añadir comportamiento a la página. Atributos: src.
<style></style>	HTML 3.2	Especifica un estilo para el documento. Puede indicarse en este elemento o enlazarse a un fichero CSS.
<title></title>	HTML 2.0	Define el título del documento, que será mostrado por el navegador como título de la ventana en que se muestra. Algunos buscadores web suelen fijarse en esta etiqueta para indexar nuestras páginas.
<meta></meta>	HTML 2.0	Especifica información adicional sobre el documento, mediante atributos como name y content. En los siguientes ejemplos hemos introducido una descripción a la página, y una lista de palabras clave, respectivamente, que facilitarán la indexación de contenidos a los motores de búsqueda. <meta name="description" content="Ejemplo para Aplicaciones Web"> <meta name="keywords" content="APW, Web, SMR, HTML, CSS">



# Estructura del HTML

- **Elemento HEAD:** Veamos algunos de ellos en detalle
- El elemento `<title>` aunque es optativo es importante ya que es un parámetro que usa los buscadores para el posicionamiento.
- También es importante la etiqueta `<meta>` porque va a determinar el tipo de caracteres a usar o para incluir metadatos o incluso refrescar la web tras 30s o que se vea bien en cualquier dispositivo. [Aquí](#) hay más detalle sobre la etiqueta.
- Respecto a la etiqueta `<link>`, se utiliza para enlazar con elementos externos al propio HTML como el fichero CSS o el icono o favicon. [Aquí](#) hay más detalle sobre la etiqueta.
- Respecto al ícono, hay que destacar que es una imagen de 16 píxeles y que puedes crear con herramientas gráficas o [aquí](#) o [aquí](#).

Práctica: Aquí tienes un ejemplo de HTML que incluye un poco de lo visto hasta ahora en la cabecera. Cambia el título de la página y busca un ícono que se pueda mostrar y a ver el cambio.

Si queréis más información sobre estas etiquetas y más cosas de HTML5 os aconsejo que visitéis el canal de [Aprendiendo Frontend](#)

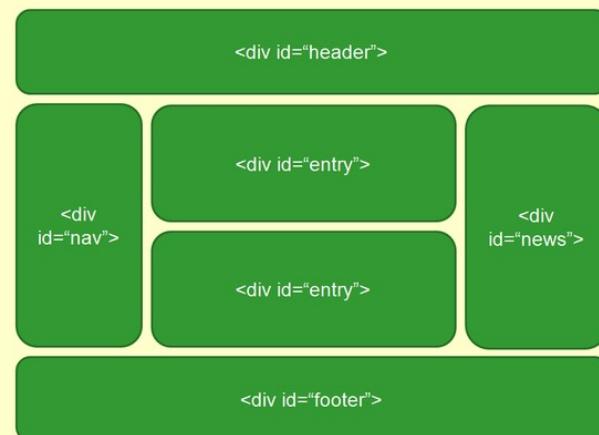
```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>My Page Title</title>
  <meta charset="iso-8859-1">
  <meta name="description" content="Ejemplo de HTML5">
  <meta name="keywords" content="HTML5, CSS3, Javascript">
  <meta http-equiv="refresh" content="30">
  <link rel="stylesheet" href="styles.css">
  <link rel="icon" type="image/x-icon" href="/images/favicon.ico">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
</head>
<body>
  <h1>This is a Heading</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```



# Estructura del HTML

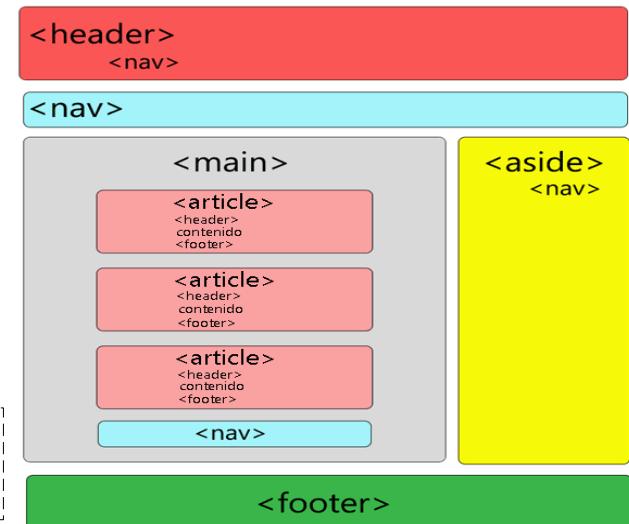
- **Elemento BODY:**

- Delimita la sección que contiene los elementos que forman la parte visible de la página HTML.  
Los elementos que la forman pueden clasificarse según la característica:
  - **Elementos de bloque:** Ocupa todo el espacio del bloque padre, puede contener otros elementos dentro y se muestran con un salto de línea antes y después. Ej: table, div, h1, header, section, embed,...
  - **Elementos de línea:** Pueden aparecer dentro de un bloque y no genera saltos de línea. Ej: a, strong, cite, script, sub,...
- Hay que destacar que con el desarrollo de HTML5, también se ha cambiado el diseño empaquetado con bloques div a una estructura ya definida con HTML5. Antes de HTML4 todo eran tablas, cada nueva versión iba mejorando la manera de maquetar la web además de añadir nuevas etiquetas.



Ejemplo página HTML5

**Vídeo:** Todo sea por mejorar el **SEO**, ¿Qué es eso?



# Estructura del HTML

- **Elemento BODY:**

- Estas son las etiquetas semánticas que estructuran un documento HTML5:
  - **<header></header>**: Para la presentación o cabecera de la página.
  - **<footer></footer>**: Para el pie de página o de una sección. Normalmente se incorpora aquí información como el autor, el copyright, la fecha, contacto, etc.
  - **<main></main>**: Es la parte principal de la página que se encuentra entre `<header>` y `<footer>` y puede contener todos o parte de los elementos siguientes. Generalmente son las secciones o artículos.
  - **<nav></nav>**: Para menús de navegación con enlaces a otras páginas del sitio o a otras webs. Su propósito es facilitar la navegación en la web.
  - **<section></section>**: Permite dividir el contenido web en secciones o como un índice de la web. Antes se hacía con div que queda relegada a cosas excepcionales.
  - **<article></article>**: Permite incluir artículos o subsecciones en otras partes de la página. Representa un componente autónomo dentro de la página y que puede repetirse en su interior. Por ejemplo, en un blog sería cada noticia y dentro hay contenido con diferentes subsecciones y artículos.
  - **<aside></aside>**: Representa un contenido de la página distinto del que lo rodea. Puede ponerse aquí contenido publicitario, otros elementos de navegación o contenido que se considera separado del contenido principal de la página. Puede estar a la derecha o izquierda de la página.
- Estas etiquetas se utilizan para estructurar la página, por lo que todas ellas son elementos de bloque en las que se pueden incluir otros elementos de bloque o en línea que veremos a continuación.
- Cabe destacar que la etiqueta **<div>** puede utilizarse pero no tiene contenido semántico al igual que **<span>**





# Estructura del HTML

- **Actividad:** Reconoce cada una de estas nuevas etiquetas de HTML5 según las definiciones anteriores.

**Práctica:** Vamos a interpretar el código que se muestra en la web  
<https://aprende-web.net/NT/html5/ejemplo1.html>

Lo descargaremos y visualizaremos en nuestro servidor y luego incorporaremos el CSS para ver la diferencia.

En el desarrollo web ten siempre presente que HTML, CSS y JavaScript son complementarios pero que no es bueno mezclar el contenido de todos en el fichero HTML.

## Three good friends



html  
The builder



css  
The artist



java script  
The wizard

**Mi Blog** ①

[principal](#) | [ayuda](#) | [contacto](#) ②

Domingo, 10 de Julio 2011 ③

**Lorem ipsum 2**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam adipiscing, purus et sagittis mollis, leo diam dapibus quam, ac varius mauris ligula ut neque. Morbi id malesuada eros. Phasellus tempus bibendum varius. Nulla facilisi. Mauris porttitor libero porta tortor facilisis eu volutpat purus porttitor. Nulla venenatis ipsum id lorem gravida in pharetra erat pharetra.

Domingo, 8 de Julio 2011

**Lorem ipsum 1** ⑥

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam adipiscing, purus et sagittis mollis, leo diam dapibus quam, ac varius mauris ligula ut neque. Morbi id malesuada eros. Phasellus tempus bibendum varius. Nulla facilisi. Mauris porttitor libero porta tortor facilisis eu volutpat purus porttitor. Nulla venenatis ipsum id lorem gravida in pharetra erat pharetra.

④

**Archivo**

> 2011 (2)  
> Julio (2)  
Lorem ipsum 2  
Lorem ipsum 1

**Acerca de**

carlos  
Ver mi Perfil >

⑤

Derechos Reservados 2010-2011



# Estructura del HTML

## Estructura de una etiqueta HTML

```
<etiqueta atributo = "valor"> contenido </etiqueta>
```

En Moodle tenéis una chuleta con las etiquetas más relevantes de HTML5 para las prácticas y actividades.

- De acuerdo con este formato podemos encontrar las que sí tienen contenido como:
  - `<h1 class="titulo2" title="este es titulo">Bienvenido a mi página web</h1>`
  - `<strong>Contenido</strong>`
- Pero también hay etiquetas que sólo tienen atributos y no contenido o incluso sólo la etiqueta como en estos casos:
  - `<img src=·"img/i1.jpg"></img> → <img src=·"img/i1.jpg" />`
  - `<hr>, <br/>`
- Nunca hay que olvidar **comentar** las cosas durante el desarrollo. HTML tiene la siguiente etiqueta para documentar que es común a XML:
  - `<!-- Esto es un comentario de ejemplo que el navegador ignorará -->`
- De cara a ser productivo, podemos agilizar el desarrollo web usando **códigos de Emmet**.
  - **Práctica:** Vamos a probar algunos de los **códigos Emmet** y ver el resultado

**Importante:** Hay que recordar que los atributos pueden ser globales o propios de la etiqueta.



# Estructura del HTML

- **Etiquetas HTML:**

## Algunos elementos de bloque

Elementos de bloque		
Elemento	Versión	Descripción
<p></p>	HTML 2.0	Introduce un párrafo en la página. Es los elementos más utilizados.
<h1></h1> ... <h6></h6>	HTML 2.0	Añade títulos de secciones a diferentes niveles: <h1> delimita los títulos de mayor nivel, el siguiente nivel es <h2> y así hasta <h6>.
<ol></ol> <ul></ul> <li></li> <dl></dl> <dt></dt> <dd></dd>	HTML 2.0	Define diferentes tipos de listas y sus elementos: <ul style="list-style-type: none"> <li>• &lt;ol&gt;&lt;/ol&gt; define listas ordenadas.</li> <li>• &lt;ul&gt;&lt;/ul&gt; define listas no ordenadas.</li> <li>• &lt;li&gt;&lt;/li&gt; define los elementos para las listas &lt;ol&gt; y &lt;ul&gt;.</li> <li>• &lt;dl&gt;&lt;/dl&gt; define una lista de definiciones.</li> <li>• &lt;dt&gt;&lt;/dt&gt; define un término en una lista &lt;dl&gt;.</li> <li>• &lt;dd&gt;&lt;/dd&gt; define la definición de un término en la lista &lt;dl&gt;.</li> </ul>
<div></div>	HTML 3.2	Define una división lógica del contenido a nivel de bloque. Se trata de un elemento genérico sin valor semántico, y se utilizan para distinguir secciones del documento con diferentes formatos o comportamientos.
<hr/>	HTML 2.0	Introduce una regla horizontal.

Elementos inline		
Elemento	Versión	Descripción
<a></a>	HTML 2.0	Permite añadir enlaces dentro del documento o a sitios web externos, mediante el atributo <code>href</code> (hypertext reference) y la URL del sitio o referencia interna. Además, mediante el atributo « <code>title</code> » podemos añadir información sobre el enlace.  <a href="www.gmail.com" title="Correo gmail">Mi Correo</a>
<abbr></abbr>	HTML 4.0	Indica el texto como una abreviatura.
<dfn></dfn>	HTML 3.2	Indica la definición en línea de un término.
<em></em>	HTML 2.0	Enfatiza un texto en cursiva.
<strong></strong>	HTML 2.0	Enfatiza un texto en negrita.
<span></span>	HTML 4.0	Describe una división lógica en línea. Un elemento sin significado semántico que distingue una sección del documento, con aspecto o comportamiento específico. Sería el equivalente, en línea, a los elementos <div> de bloque.
 	HTML 2.0	Fuerza un retorno de carro que pasa a la línea siguiente.
<cite></cite>	HTML 2.0	Referencia a una cita en el documento.
<del></del>	HTML 4.0	Define un texto como eliminado. Genialmente se muestra como tachado.
<ins></ins>	HTML 4.0	Marca el texto como insertado. Se suele mostrar como subrayado.
<sub></sub>	HTML 3.2	Marca el texto como subíndice ( <code>sub</code> ) o superíndice ( <code>sup</code> ).
<sup></sup>		

**¡Importante!** Hay etiquetas heredadas de versiones anteriores pero es importante que uses las nuevas ya que agregan semántica y accesibilidad como figure frente a usar solo img. Aquí verás ejemplos de uso de cada una.

## Algunos elementos inline



# Estructura del HTML

## • Empecemos por escribir texto

- Para escribir párrafos de texto en HTML podemos hacer lo siguiente en el body:

< p > estas etiquetas definen un párrafo </ p >

- Dentro de las etiquetas podremos incluir otras para dar formato al texto como:

< p > Párrafo con contenido en < strong > negrita y < em > cursiva </ em > </ strong > </ p >

Formato al texto	Etiqueta HTML
CABECERAS	< h1 >, < h2 >, < h3 >, < h4 >, < h5 >, < h6 >
<b>texto en negrita</b>	< strong > texto en negrita < / strong >
<i>texto en cursiva</i>	< em > texto en cursiva < / em > o < cite > texto < / cite >
<u>texto subrayado</u>	< u > texto subrayado < / u >
texto en superíndice	< sup > superíndice < / sup >
texto en subíndice	< sub > subíndice < / sub >
Texto en letra pequeña	< small > texto < / small >
Tachar texto	< del > texto < / del >
Texto resaltado	< mark > texto < / mark >
Abreviatura	< abbr title = "texto" > SIGLAS < / abbr >
Definiciones	< dfn title = "texto" > palabra < / dfn >
Código informático	< code > código < / code >

Hay muchos modificadores de texto como son los siguientes:

**Práctica:** Vamos a ver el resultado de estos códigos en un ejemplo desde VSCode



# Estructura del HTML

- **Elementos para indicar secciones de tipo mostrar/ocultar**
  - Se trata de secciones que están dentro de un elemento llamado **details**. Dentro de ese elemento se puede poner una sección de tipo **summary** la cual muestra un texto de resumen de la sección. Al hacer clic en el apartado **summary**, se muestra todo el contenido de la sección. Veremos que también se puede hacer esto con bloques div con JavaScript
  - **Práctica:** Realiza este ejemplo para ver el resultado

```
<body>
<details>
  <summary>
    La ciudad de Palencia
  </summary>
  <p>Localidad de 80.000 habitantes situada entre los Valles de Cerrato y la Tierra de Campos en plena meseta castellana</p>
  <p>Más información en
    <a href="https://es.wikipedia.org/wiki/Palencia">
      Wikipedia-Ciudad de Palencia
    </a>
  </p>
</details>
</body>
```

▼ La ciudad de Palencia

Localidad de 80.000 habitantes situada entre los Valles de Cerrato y la Tierra de Campos en plena meseta castellana

Más información en [Wikipedia-Ciudad de Palencia](https://es.wikipedia.org/wiki/Palencia)



# Estructura del HTML

- **Listas en HTML**
- Las listas se usan para hacer una enumeración de algo, existen dos tipos de listas:
  - Ordenadas, son aquellas que se indican con letras (a,b,c) o números (1,2,3, I, II). En HTML se corresponden con la etiqueta <ol>
  - Desordenadas, son las que se indican con iconos como círculo, cuadrado, guión,... y que se corresponden con la etiqueta <ul>
  - Listas de términos: Son las que se especifican con <dl> y dentro tienen <dt>palabra</dt> y su <dd>significado</dd>

Puedes encontrar más detalle de cada uno y otros [aquí](#)

**Práctica:** Realiza en un fichero html estos códigos a ver el resultado que muestra cada uno.

```
<ol>
  <li>Elemento 1</li>
  <li>Elemento 2</li>
  <li>Elemento 3</li>
  <li>Elemento 4</li>
</ol>

<ol start="20">
  <li>Elemento 1</li>
  <li>Elemento 2</li>
  <li>Elemento 3</li>
  <li>Elemento 4</li>
</ol>
```

```
34 <ol type="a">
35   <li>Elemento 1</li>
36   <li>Elemento 2</li>
37   <li>Elemento 3</li>
38   <li>Elemento 4</li>
39 </ol>
40
41 <ol type="I">
42   <li>Elemento 1</li>
43   <li>Elemento 2</li>
44   <li>Elemento 3</li>
45   <li>Elemento 4</li>
46 </ol>
```

```
1 <ol reversed>
2   <li>Elemento 1</li>
3   <li>Elemento 2</li>
4   <li>Elemento 3</li>
5   <li>Elemento 4</li>
6 </ol>
7
8 <ol reversed="reversed">
9   <li>Elemento 1</li>
10  <li>Elemento 2</li>
11  <li>Elemento 3</li>
12  <li>Elemento 4</li>
13 </ol>
```

```
<ul>
  <li>Elemento 1</li>
  <li>Elemento 2</li>
  <li>Elemento 3</li>
  <li>Elemento 4</li>
</ul>

<ul type="circle">
  <li>Elemento 1 iAtención type deprecated!</li>
  <li>Elemento 2 iAtención type deprecated!</li>
  <li>Elemento 3 iAtención type deprecated!</li>
  <li>Elemento 4 iAtención type deprecated!</li>
</ul>

<ul type="square">
  <li>Elemento 1 iAtención type deprecated!</li>
  <li>Elemento 2 iAtención type deprecated!</li>
  <li>Elemento 3 iAtención type deprecated!</li>
  <li>Elemento 4 iAtención type deprecated!</li>
</ul>
```

# Estructura del HTML

## • Hiperenlaces en HTML

- No hay que olvidar que HTML se creó para enlazar documentos en Internet. Para crear hiperenlaces en nuestra página web usamos la etiqueta:  
`<a href="url">Texto del enlace</a>`
- Por ejemplo si queremos hacer un enlace al buscador **google** para que se abra esta página, pondremos lo siguiente:
  - `<a href="http://www.google.es">Google</a>` → No olvides el protocolo http o el que corresponda!!
  - Si el ejemplo anterior lo pulsamos se abrirá la página en la misma que estemos. Si queremos que al pulsar nos abra una pestaña nueva con el contenido tendremos que hacer lo siguiente:  
`<a href="http://www.google.es" target= "_blank">Google</a>` → Hay más atributos útiles [aquí](#)
  - En caso de querer enlazar a páginas propias, habría que colocar la ruta relativa a donde esté la página web. El enlace puede ser una página o un documento. En este segundo caso se iniciará la descarga. Un ejemplo será considerando que está la página en el mismo directorio:  
`<a href="pagina.html">Página</a>` (Sin http se refiere a una página nuestra)

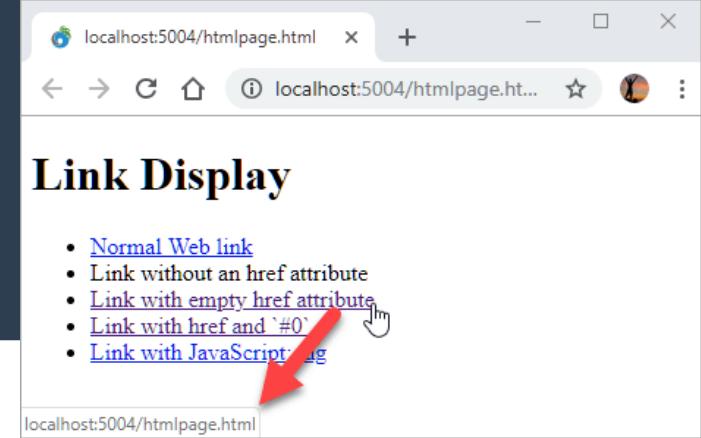
Si se hace a la propia página sería `con href="#"` o para una parte que esté con id con `href="#id"` (será un ancla)

- También podríamos usar el hiperenlace para enviar un correo.

- `<a href="mailto:domingolopez.instituto@gmail.com">Enviar mail.</a>`

Con HTML5, se suele usar esto dentro de la etiqueta `<address>`

**Práctica:** Realiza un ejemplo similar a la figura con 4 enlaces: interno, correo, externo y externo en pestaña (por ejemplo tu blog)



# Estructura del HTML

## • Hiperenlaces en HTML

- Caso de los **anclas**: Este tipo de enlaces también son importantes cuando queremos facilitar la navegación a una sección concreta de la página, por ejemplo al final del artículo, al principio... a través de un menú
- Ejemplo:

Vamos a crear una página web HTML5 donde incluyamos en el body...

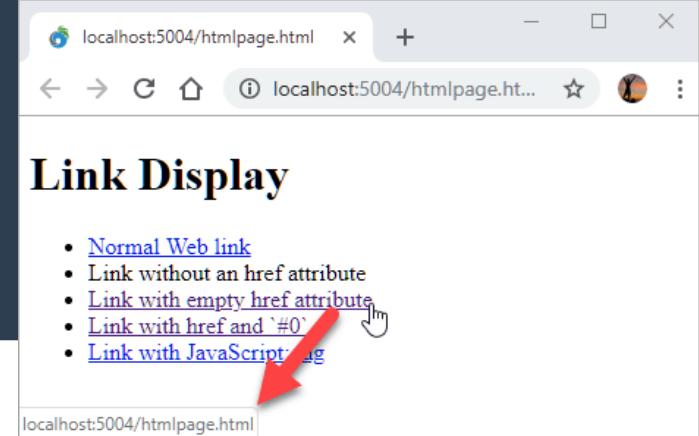
```
<nav>
  <ul>
    <li><h1><a href="#ir-a-principiantes">Principiantes</a></h1></li>
    <li><h1><a href="#ir-a-intermedios">Intermedios</a></h1></li>
    <li><h1><a href="manual.html#avanzados">Avanzados</a></h1></li>
  </ul>
</nav>
```

En este fichero

```
<article>
  <h1 id="ir-a-principiantes">Principiantes</h1>
  <p>Texto introducción</p>
</article>
<article>
  <h1 id="ir-a-intermedios">Intermedios</h1>
  <p>Texto desarrollo</p>
</article>
```

**En el fichero manual.html tiene que haber...**

```
<article>
  <h1 id="avanzados">Avanzados</h1>
  <p>Texto fin</p>
</article>
```



**Actividad:** Elabora un documento HTML que partiendo de este ejemplo me permita moverme en la página web y que los artículos incluyan un enlace de volver arriba. Copia y pega un contenido largo para que se vea el efecto. Ej: Lorem Ipsun...



# Estructura del HTML

## • Tablas en HTML:

- La etiqueta de bloque (<table>) representa una tabla y permite crear una estructura de datos agrupados en filas <tr> y columnas <td>.
- Para la cabecera de la tabla se suele usar <th> en lugar de <td>

<table>

<td></td>	<td></td>	<td></td>

</tr>

</tr>

</tr>

</tr>

</table>

```
<!DOCTYPE html>
<html>
<body>
<table border="1" style="width:100%">
<tr>
  <th>Nombre</th>
  <th>Apellido</th>
  <th>Nota</th>
</tr>
<tr>
  <td>Carla</td>
  <td>Montes</td>
  <td>9</td>
</tr>
<tr>
  <td>Juan</td>
  <td>León</td>
  <td>8</td>
</tr>
</table>
</body>
</html>
```

Nombre	Apellido	Nota
Carla	Montes	9
Juan	León	8

Al definir la tabla podremos indicar atributos como el ancho, borde o estilos en la tabla o sus componentes td y tr. Sin embargo, es mejor dejar aspectos visuales para CSS.

**Práctica:** Prueba este ejemplo y los siguientes en el mismo fichero HTML  
Después incluiremos las nuevas etiquetas **thead**, **tbody**, **tfoot** de HTML5



# Estructura del HTML

- **Tablas en HTML:**

- Las tablas permiten múltiples configuraciones cuando una celda puede ocupar varias filas o varias columnas. Para ello se emplean los atributos **rowspan** (filas) y **colspan** (columnas) de forma apropiada.

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
}
</style>
</head>
<body>
<h1>The td rowspan attribute</h1>
<table>
<tr>
  <th>Month</th>
  <th>Savings</th>
  <th>Savings for holiday!</th>
</tr>
<tr>
  <td>January</td>
  <td>$100</td>
  <td rowspan="2">$50</td>
</tr>
<tr>
  <td>February</td>
  <td>$80</td>
</tr>
</table>
</body>
</html>
```

Month	Savings	Savings for holiday!
January	\$100	
February	\$80	

```
<table width=200>
<tr>
  <th>Month</th>
  <th>Savings</th>
  <th>Sum</th>
</tr>
<tr>
  <td>January</td>
  <td>$100</td>
  <td rowspan="2">Sum: $180</td>
</tr>
<tr>
  <td>February</td>
  <td>$80</td>
  </td>
</tr>
<tr>
  <td colspan="2" rowspan="2">Total:</td>
  <td>Sum: $180</td>
</tr>
</table>
```

Esto NO es HTML, es CSS. Manténlo en los ejemplos y actividades para que se vean bien las celdas de la tabla

```
<table width=50%>
<tr>
  <th colspan="3">2022</th>
</tr>
<tr>
  <td>&ampnbsp</td>
  <td>&ampnbsp</td>
  <td>&ampnbsp</td>
</tr>
<tr>
  <th colspan="2" rowspan="2">Fiesta</th>
  <td>&ampnbsp</td>
</tr>
<tr>
  <td>&ampnbsp</td>
  <td>&ampnbsp</td>
</tr>
<tr>
  <td colspan="3">2022</td>
</tr>
<tr>
  <td colspan="3">Fiesta</td>
</tr>
```

2022		
Fiesta		



# Estructura del HTML

## • Tablas en HTML:

- Ahora con HTML5 aparecen otras etiquetas semánticas como son:
  - thead. Sirve para indicar las filas que forman la cabecera de la tabla
  - tfoot. Indica el pie de la tabla
  - tbody. Indica el cuerpo de la tabla
- Ejemplo:

**Actividad:** Realiza los siguientes ejemplos de tablas en HTML5 y tu horario de clase de 1º ASIR. NO HAY QUE COLORAR LAS CELDAS

No	problem	for	a web
	professional		
you			as
will			
in	see	a few	minutes!

1.1	1.2
31.	2.2
4.1	

```
<table border="1">
  <caption>Ventas por secciones</caption>
  <thead>
    <tr>
      <td>&nbsp;</td>
      <td>Hardware</td>
      <td>Software</td>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <th>Total</th>
      <th>25000</th>
      <th>22000</th>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <th>Enero</th>
      <td>12000</td>
      <td>15000</td>
    </tr>
    <tr>
      <th>Febrero</th>
      <td>13000</td>
      <td>9000</td>
    </tr>
  </tbody>
</table>
```

Muy útil para dejar un nombre de tabla



# Estructura del HTML

- **Tablas en HTML:**

- Recuerda que cada elemento puede ser a su vez ser un contenedor. No tiene por qué ser texto siempre y podría ser otra tabla como en el ejemplo:

						<table border="1"><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>													

**Práctica:** Emplea las etiquetas de table para crear este formato de tabla. Pensad, que así se construía antes una página web antes de usar HTML4 o HTML5.



# Estructura del HTML

- **Incluir imágenes en HTML:**

- Si queremos incluir una **imagen** en nuestro contenido, tendremos que hacer uso de la etiqueta FIGURE y dentro acceder a la imagen a mostrar con img. Puedes encontrar más detalle de su uso [aquí](#).

```
<figure>
    
    <figcaption>
        Esta es la imagen del primer mensaje
    </figcaption>
</figure>
```

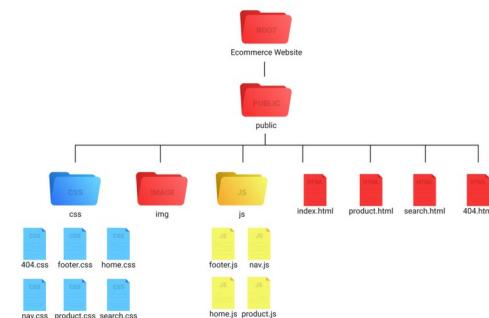
I **Práctica:** Vamos a probar el ejemplo de la web

- Recuerda que la ruta a “myimagen.jpg” se puede hacer por ruta absoluta, ruta relativa o una URL como el ejemplo. En la ruta absoluta estará TODA la dirección desde C:\, y como ruta relativa desde la carpeta donde está el fichero html. Se suelen utilizar las rutas relativas.

- Existen varias propiedades para modificar características de la imagen, por ejemplo:

- ancho de la imagen -- **width**
- alto de la imagen -- **height**
- título -- **title**
- texto alternativo – **alt**

Hasta ahora no se ha accedido a  
ficheros internos pero recuerda  
mantener una estructura web adecuada





# Estructura del HTML

## • Incluir imágenes en HTML:

- Sin embargo, no hay que olvidar que hay muchos formatos de imagen (jpg, gif, png, wepp, svg,...) y puede que el navegador no sepa interpretar ese formato correctamente. Por ejemplo, el formato webp sólo lo interpreta algunos navegadores.
- En este caso, también podemos optar por la etiqueta **picture** que intenta mostrar una u otra imagen según pueda ser procesada o no por el navegador o por el tamaño que puede visualizar en píxeles (media="max-width:600px")

```
<picture>
  <source srcset="logo.webp" type="image/webp">
  <source srcset="logo.png" type="image/png">
  
</picture>
```

- No olvides que puedes mezclar definiciones de elementos HTML para dar valor semántico y ser más utilizable. Ej:

Ten en cuenta que muchos componentes permiten incluir otros como una imagen pero que habrá que adaptar el tamaño con width y height. Preferiblemente esto se debe hacer desde CSS.

Algo similar a esto usaremos en RWD para mostrar imágenes con media. Vamos a probar esto y veremos que no siempre hay que usar CSS. ([enlace](#))

Si quiero varias imágenes dentro de la misma estructura, bastará con incluir una tras otra en un bloque div

```
<figure>
  <picture>
    <source ...>
    <source ...>
    <img..>
  </picture>
  <figcaption>...</figcaption>
</figure>
```



# Estructura del HTML

- **Incluir imágenes en HTML:**
- ¿Y cómo incluyo los **iconos** que aparecen en las webs? Muchos de ellos **NO** son img
  - Aquí puedes encontrar ejemplos de uso para incluir multitud de iconos que ves en la web. Se pueden mezclar iconos de diferentes orígenes o colocar los tuyos propios.
  - Si utilizas <https://fontawesome.com/> antes tendrás que registrarte para tener el **código** que debes incluir en la cabecera:
    - <script src="https://kit.fontawesome.com/**yourcode**.js" crossorigin="anonymous"></script>
  - Si quieres usar los de Google en <https://fonts.google.com/icons> tendrás que agregar lo siguiente:
    - <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

## Ejemplos: Font-Awesome:

```
<i class="fa-solid fa-user"></i>
<i class="fa-brands fa-facebook-f"></i>
<i class="fa-duotone fa-camera-retro"></i>
```

## Ejemplos: Google Icons:

```
<i class="material-icons">home</i>
<i class="material-icons">favorite</i>
<i class="material-icons">computer</i>
```

**Práctica:** Vamos a incluir algunos iconos y la imagen para ver la diferencia.

## Ventaja de esto:

Se considera como texto a la hora de hacerle CSS. Si queremos cambiar el tamaño respecto de su contenedor, tendremos que incluir un tercer parámetro en class como fa-10x que lo aumenta 10 veces. Ej: <i class="fas fa-thumbs-up fa-3x"></i> Eso sí, al principio tendrás que buscar los más apropiados a tu web y no todos los iconos son gratuitos.



# Estructura del HTML

- **Actividad resumen que incluirás en la tarea a subir a Moodle:**
    - Empleando HTML5 realiza un diseño similar al siguiente
- Ejercicio práctico con Tablas**

					
<b>Capacidad</b>	2GB	4GB	8GB	32GB	2GB
<b>Colores</b>	Blanco	<ul style="list-style-type: none"><li>• Negro</li><li>• Rosa</li><li>• Dorado</li></ul>	Negro	<ul style="list-style-type: none"><li>• Negro</li><li>• Blanco</li></ul>	
<b>Pantalla</b>	3 pulgadas			7 pulgadas	
<b>Tiempo de Carga</b>	1 hora			5 horas	
				30 minutos para 75%	

La primera columna serán `<th>` y toda la tabla estará en `<tbody>`  
El resto de campos en negrita tendrás que ponerlos tú con formato de texto.

Tendrás que buscar tú la foto y colocar el width y height apropiado



# Estructura del HTML

- **Mapa de enlaces en imágenes en HTML:**

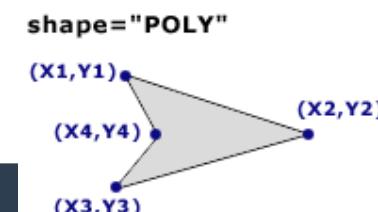
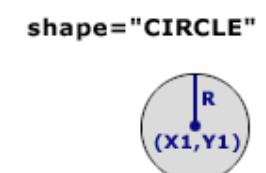
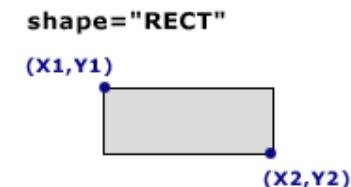
- El elemento **map**, junto con img y area, crea un mapa de imagen. Los mapas de imagen permiten a los autores definir secciones en una imagen y, opcionalmente, convertirlas en hipervínculos que apuntan a otros recursos. Estos mapas suelen formarse como rectángulos, círculos o polígonos.
- Se trata de una técnica en desuso debido a la mayor potencia que añade JavaScript para realizar acciones interactivas sobre imágenes y a la mayor facilidad que poseen las imágenes SVG para detectar áreas interactivas. No obstante, crear mapas de imágenes mediante elementos map sigue siendo una técnica compatible con todos los navegadores del mercado por lo que aun se pueden observar páginas web de escritorio en las que se aplica esta técnica.
- Ejemplo de código que usa map y las formas shape:

<p>Pincha en cada planeta para ver más detalle sobre él:</p>

```

```

```
<map name="planetmap">
  <area shape="rect" coords="0,160,90,275" alt="Sol" href="https://es.wikipedia.org/wiki/Sol">
  <area shape="circle" coords="120,220,5" alt="Mercurio"
  href="https://es.wikipedia.org/wiki/Mercurio_(planeta)">
  <area shape="circle" coords="170,220,20" alt="Venus" href="https://es.wikipedia.org/wiki/Venus_(planeta)">
  <area shape="circle" coords="245,220,20" alt="Tierra" href="https://es.wikipedia.org/wiki/Tierra">
</map>
```

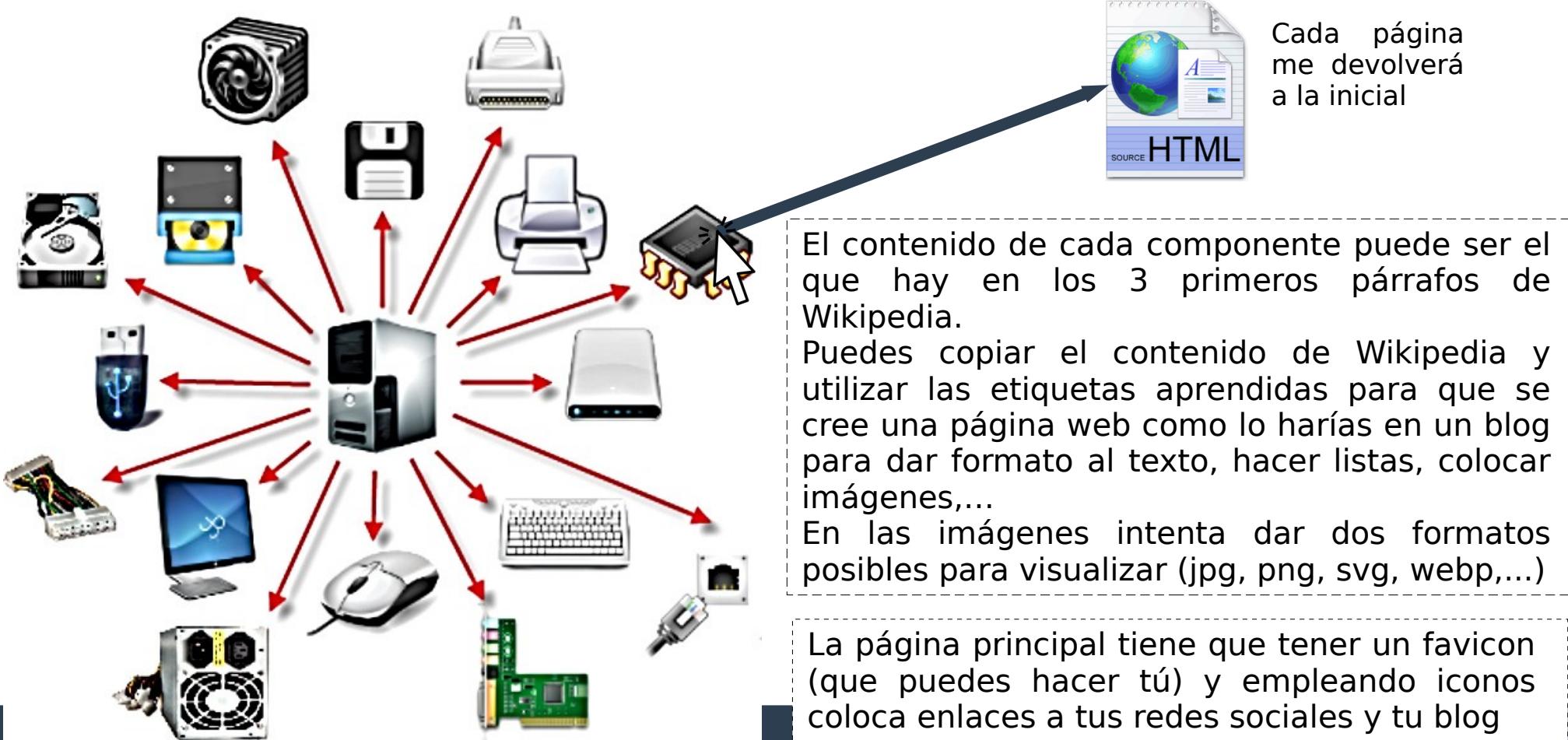


**Práctica:** Vamos a hacer el caso de un map con el sistema solar.  
Entenderéis por qué no se utiliza de forma masiva



# Estructura del HTML

- **Tarea:** Apoyándote en los contenidos de la materia de Fundamentos de Hardware, realiza páginas web donde la primera sea una foto de un ordenador con componentes y nos lleve a esas páginas desde la principal index.html que contiene la siguiente imagen y utiliza mapas de enlaces.





# Estructura del HTML

- **Formularios en HTML:**

- Hasta ahora hemos visto la forma en la que el HTML gestiona y muestra la información, esencialmente mediante texto, imágenes, listas, tablas y enlaces. Nos queda por ver de qué forma podemos intercambiar información con el usuario. Esto se hace con los formularios
- Un formulario es una caja de texto y botones que podemos encontrar en muchas páginas web. Son muy utilizados para realizar búsquedas o bien para introducir datos personales por ejemplo en sitios de comercio electrónico.
- En HTML se genera un formulario con la etiqueta **<form>** y dentro se definen los diferentes elementos de un formulario que pueden ser: botones, listas desplegables, cuadros de texto, etc, también conocidos como “campos de formulario”. Puedes ver más detalle de su estructura y atributos [aquí](#) o [aquí](#) con ejemplos.
- Siempre al final del formulario habrá un botón “Enviar” que permite enviar los datos introducidos con el método GET o POST a otra aplicación o página que los procesa. Ese es el caso de PHP y que al tratarse de BackEnd NO veremos en esta materia.



# Estructura del HTML

- **Formulario en HTML:**

- Este sería un código de un formulario que incluye la mayoría de componentes:

```
<form action="../../form-result.php" method="get" target="_blank">
<fieldset>
<legend>Información básica</legend>
<p><label>Nombre completo: <input type="text" name="nombrecompleto"></label></p>
<p><label>Fecha de nacimiento: <input type="date" name="fechadenacimiento"></label></p>
<p>Gender:
<label><input type="radio" name="genero" value="masculino"> Masculino</label>
<label><input type="radio" name="genero" value="femenino"> Femenino</label>
</p>
<p><label>Dirección: <input type="text" name="direccion"></label></p>
<p><label>Número telefónico: <input type="tel" name="telefono"></label></p>
</fieldset>

Fijaros que hay diferentes type que llevan formato como
fecha o teléfono. Hay otros como password o con patrones.

<label for="texto"> Escribe el nº de serie </label>
<input type="text" pattern="[A-Z]{5}[0-9]{3}" id="texto" name="texto"
placeholder="5 letras y tres números">

<label for="nombre">Escriba su nombre</label>
<input type="password" name="nombre" id="identificador">
```

Aquí le indicamos la acción del botón, el método y si se muestra en otra página. Si lo obviamos no hará nada

```
<input type="submit" value="Enviar datos">
<input type="reset" value="Restaurar formulario">
</p>
</form>
```



# Estructura del HTML

- **Formulario en HTML:**

- Esta sería el resultado en el navegador:

Información básica

Nombre completo:  → Input type="text"

Fecha de nacimiento:  → Input type="date"

Gender:  Masculino  Femenino → Input type="radio"

Dirección:

Número telefónico:  → Input type="tel"

Información extra

Interests:  Libros  Películas  Videojuegos → Input type="checkbox"

Color favorito:  → Input type="color"

→ Input type="submit"

→ Input type="reset"

En cuanto pulsemos enviar enviará la información al fichero form-result.php con método GET



# Estructura del HTML

- **¿Cómo funciona un formulario?:**

- Un formulario es un contenedor de controles que sirven para que el usuario introduzca datos.
- Cada control posee un nombre al cual se le asigna un valor, el cual se corresponde con lo que el usuario introduce en el formulario. De este modo, cuando se envían los datos del formulario (normalmente a través de un botón), se envían todos los nombres y valores de los controles del formulario hacia el servidor o web que los procesa.
- El servidor procesa estos datos, pero de forma opaca para el usuario. No se transmite el código que hace posible el proceso, sino el resultado del mismo. Si no quisiera usar un fichero externo PHP con el action, tendría que incluir sólo los input y gestionar submit con JavaScript.
- Al usuario le llegará una respuesta, relacionada con los datos que se han procesado, normalmente en forma de página web.
- Este proceso de envío de datos al servidor se puede llevar a cabo con los métodos GET y POST



# Estructura del HTML

- **¿Cómo funciona un formulario?:**

- **Paso de datos mediante GET:** Al indicar method="GET", o si no se especifica, en form estamos indicando que se enviará la URL al servidor con este formato:

`http://urlpágina?var1=valor1&var2=valor2&....`

- Es decir, se envían pares entre el nombre de la variable (en el formulario asignada a través del atributo name) y el valor que se le dio en el formulario. Cuando un control del formulario se queda sin valor, entonces el nombre del control se queda sin definir (el servidor no recibiría ningún dato con ese nombre).
    - El principal problema de este método es la seguridad, ya que los datos viajan con la propia URL sin cifrar. Por ejemplo, si se hiciera así el login se verían las contraseñas.

- **Paso de parámetros mediante POST:** Para solventar este fallo de seguridad, el paso por POST funciona exactamente igual que GET pero se pasan los (nombre, valor) dentro del paquete de datos y no en la URL.



# Estructura del HTML

- **Estructura de un formulario en HTML:**

- De esta forma, todo formulario tiene que empezar de la siguiente forma:

```
<form action="página/servicio.php" method="GET o POST">  
...  
</form>
```

- Hay otros atributos interesantes para el formulario como los siguientes:

atributo	significado
accept-charset	Codificación de caracteres que se utilizará para pasar los contenidos del formulario, por ejemplo ISO-8859-1. Por defecto se usa la codificación de la propia página web.
autocomplete	Puede tomar los valores <b>on</b> u <b>off</b> . Si se activa esta opción (con valor <b>on</b> ), el navegador automáticamente completará los datos del formulario basándose en los valores para esos mismos controles que el usuario haya llenado en otros formularios. Este atributo está disponible también para cada control (así podremos indicar el auto completado en unos controles y en otros no). No funciona en el navegador <b>Opera</b> .
enctype	Solo es válido para el método POST. Indica el formato en el que los datos del formulario son enviados. Nunca se suele usar. Posibles valores: <ul style="list-style-type: none"><li>▪ <b>application/x-www-form-urlencoded</b>. Formato predeterminado para el envío de los datos.</li><li>▪ <b>multipart/form-data</b>. Formato de dato binario, solo se usa si con el formulario se envía un archivo.</li><li>▪ <b>text/plain</b>. Texto plano, compatible con antiguos programas de recepción de datos de formulario (por ejemplo viejos programas CGI en el servidor).</li></ul>
novalidate	Los navegadores compatibles con HTML 5 tienen la capacidad de validar algunos datos (por ejemplo en los cuadros numéricos se valida que el usuario realmente ha escrito un número). Si usamos este atributo (que no tiene valor asociado), esas validaciones no se harán
target	Es el mismo atributo que en la etiqueta <b>a</b> . Si se usa con valor <b>_blank</b> , entonces la respuesta al formulario se da en otra ventana.

Aquí puedes ver un ejemplo de cómo se procesa el action

Vamos a ver cómo diseñamos el interior del formulario como **input** y sus correspondientes etiquetas **label** si queremos mostrar información al usuario.



# Estructura del HTML

- **Estructura de un formulario en HTML:**

- El elemento más importante del formulario es `<input>` , y tiene distintas variaciones dependiendo del tipo de atributo. A continuación se detallan los más importantes:
  - `<input type="text">` : define un campo para introducir texto.
  - `<input type="password">`: igual al anterior salvo que los caracteres se enmascaran por asteriscos o círculos.
  - `<input type="radio">`: define un botón de tipo radio que solo permite elegir una de las opciones.
  - `<input type="checkbox">`: define una serie de casillas pudiendo elegir desde cero a más opciones.
  - `<input type="date">`: en HTML5 muestra el icono del mapa como forma de introducir la fecha.
  - `<input type="email" name="email">` en HTML5 valida que hemos escrito el email siguiendo el patrón correcto.
  - `<input type="submit">`: define un botón que permite enviar a la página especificada en el action todos los valores recogidos en el formulario por los distintos campos. También está el type “reset” para borrar formulario y “button” para crear nuestra propia acción JavaScript. En el botón podremos indicar una imagen de icono.



# Estructura del HTML

- **Estructura de un formulario en HTML:**

Ahora bien, type no es el único atributo importante para el elemento input. Hay algunos más:

También podemos indicar el formato en el que queremos el contenido con el atributo **pattern** o mostrar al usuario cómo lo queremos con **placeholder**

atributo	significado
<b>type</b>	Indica el tipo de control. Sus posibles valores se explican en las siguientes secciones del manual
<b>name</b>	Atributo fundamental que da un nombre al control. Este nombre es el que se le pasa al servicio receptor de los datos del formulario; realmente se le pasa el nombre y el valor que el usuario ha introducido en el control.
<b>size</b>	Tamaño, en caracteres, que tendrá el cuadro (especialmente útil en los cuadros de texto y numéricos)
<b>value</b>	Permite dar un valor al elemento. Antes de que el usuario rellene datos en él, aparecerá el valor indicado con este atributo.
<b>disabled</b>	El control aparecerá deshabilitado para su edición.
<b>autocomplete</b>	Permite que el navegador rellene automáticamente el contenido del control en base a la información que posee del usuario. Los valores posibles son on u off.
<b>autofocus</b>	Hace que elemento obtenga el foco (el foco hace que el control del formulario que lo tenga sea el que recibe las pulsaciones de teclado del usuario) en cuanto cargue la página.
<b>disabled</b>	El control estará deshabilitado
<b>form</b>	Recibe el identificador del formulario al que pertenece el control. Se usa cuando el elemento está fuera de la etiqueta form . Internet Explorer no es compatible con este atributo. Es un atributo de HTML 5
<b>formnovalidate</b>	Hace que el control no sea validado cuando se envíen los datos
<b>maxlength</b>	Permite indicar el máximo número de caracteres que se admitirán al llenar el control
<b>id</b>	<p>Este atributo es el común a todos los elementos HTML. En el caso de los formularios su uso no era habitual ya que necesitan del uso de name que ya se puede entender que identifica a cada control.</p> <p>Sin embargo se aconseja usar siempre id en los controles por estas razones:</p> <ul style="list-style-type: none"><li>▪ El atributo <b>name</b> ya hace bastantes años que se quería eliminar. Aunque esta no es razón para ello ya que, por ahora, el atributo name es la forma de pasar los datos de un control de formulario al servicio que los recibe</li><li>▪ La nueva etiqueta <b>label</b> (introducida en HTML 5) que sirve como texto de acompañamiento de los controles de un formulario, requiere que el control al que etiqueta esté identificado.</li><li>▪ La manera más habitual para acceder desde JavaScript a un control del formulario es a través de su identificador. Como es habitual usar JavaScript cuando se usan los formularios, es un excelente hábito indicar siempre valor para el atributo <b>id</b>.</li></ul>
<b>readonly</b>	Indica que el control es de <i>solo lectura</i> ; es decir, no se podrá escribir en él.
<b>required</b>	Atributo de HTML 5 que hace que el elemento sea de obligado llenado; es decir que no se puede dejar vacío, el usuario tendrá que darle algún valor.



# Estructura del HTML

- **Estructura de un formulario en HTML:**

- Ejemplos de códigos input según su atributo type: Las etiquetas label colocan el texto asociado a cada control

```
<form action="servicio.php"
      method="get">
  <label for="texto">
    Escribe el nº de serie
  </label>
  <input type="text" pattern="[A-Z]{5}[0-9]{3}" id="texto"
         name="texto" placeholder="5 letras y tres números">
  <input type="submit" value="enviar"><br>
</form>
```

```
<form action="servicio.php" method="get">
  <p>Sexo:</p>
  <input type="radio" name="sexo" id="hombre" value="h">
    <label for="hombre">Hombre</label><br>
  <input type="radio" name="sexo" id="mujer" value="m" checked>
    <label for="mujer">Mujer</label> <br>
</form>
```

```
<input type="number" name="numero" min="20">
```

Escribe el nº de serie

5 letras y tres números

enviar

Le podemos decir el número de filas y columnas como tamaño

Con placeholder se indica al usuario lo que quieres poner en el input pero realmente no hay nada, NO es lo mismo que value que sí pone un valor y que se enviará con él si no lo cambiamos.

Sexo:

Hombre  
 Mujer

¿Cuál es su edad? 34

Le podemos indicar mínimo, máximo y paso para elegir

Si quieres botones personalizados, como añadir una imagen, lo tendrás que hacer así:

```
<button type="reset">
  
    Borrar
  </button>
```



También puedes poner un ícono como:  
*</i>*



# Estructura del HTML

- **Estructura de un formulario en HTML:**

- Ejemplos de códigos input según su atributo type:

```
<input type="color" name="colorElegido" value="#ff0000">
```

→ Elija un color:

Indique su nivel (de 1 a 10, de 3 en 3):

```
<input type="range" name="rango" min="1" max="10" step="3">
```

→ Indique su nivel (de 1 a 10, de 3 en 3):

```
<select name="menu">
  <option value="1">Uno</option>
  <option value="2">Dos</option>
  <option value="3" selected>Tres</option>
</select>
```

→ Elija una opción:

El valor asociado es el del contenido salvo que se indique value. Con size le podrías indicar el tamaño visible de la selección. También puedes hacer grupos con [optgroup](#). Actualmente HTML5 permite crear esto también en cuadros de texto con [datalist](#)

```
<input type="checkbox" name="casilla" checked>Casilla 1
```

→ Casilla 1

Fecha: <input type="date" name="fecha">

→ Fecha:

Hora: <input type="time" name="hora">

→ Hora:

```
<fieldset>
  <legend>Datos personales</legend>
    <input>...
</fieldset>
```

Datos personales

Hombre  
 Mujer

Estado civil: Casado

Podemos agrupar varios de estos elementos bajo un [fieldset](#) que hará un recuadro y con legend le pondrá un nombre. Estéticamente es mejor agrupar que poner un campo tras otro.



# Estructura del HTML

- **Estructura de un formulario en HTML:**

- Controles avanzados:

- Ficheros: Permite buscar ficheros según el formato en tu equipo. Un ejemplo que acepta sólo imágenes o pdf es:

```
<input type="file" accept="image/*,.pdf" />
```

- Output: Se utiliza para mostrar resultados procedentes de cálculos sobre otros controles del formulario.

Aquí suele usarse JavaScript

```
<form oninput="total.value=parseInt(n1.value)+parseInt(n2.value)">  
  <input type="number" id="n1" name="n1" value="0">  
  +<input type="number" id="n2" name="n2" value="0">  
  =<output name="total" for="n1 n2" id="total"></output>  
</form>
```

- Barra de estado: Se trata de un tipo de barra que sirven para indicar de forma gráfica un determinado valor fracción o parte de algo. No hay que confundirla con la barra de progreso.

```
<meter value="138" min="1" max="200"  
high="150" low="60" >138</meter>
```

```
<progress max="100" value="35">35%</progress>
```





# Estructura del HTML

- Práctica 1: Realiza el siguiente formato de formulario en HTML5

Nombre:

Correo Electrónico:

URL:  Escribe la URL de tu página web

Fecha:  dd/mm/aaaa

Tiempo:  - : - -

Fecha y hora de nacimiento:

Mes:  - - - - de  - - - -

Semana:  Semana - - -, - - - -

Número (min -10, max 10):  0

Intervalo (min 0, max 10):  0

Teléfono:

Término de búsqueda:

Color Favorito:

**Debate:** ¿cómo plantearías este?

Sign up

Username

E-mail

OR

Password

Retype password

Si pones los campos como required y con tipo concreto verás que te pide que tengan ese formato al pulsar Submit!

Por supuesto, CSS evitará el uso de table como en este ejemplo: [enlace](#)



# Estructura del HTML

- **Práctica 2: Realiza el siguiente formato de formulario en HTML5**

- Verás que dentro del formulario también puede haber estructuras vistas antes en HTML5

Datos personales

1. Nombre   
2. Email   
3. Teléfono

Dirección de envío

Avenida....

- Dirección
- Código Postal
- País

Card details

Card type

1.  VISA  
 AmEx  
 Mastercard

2. Número de tarjeta   
3. Código de seguridad   
4. Propietario

Usaremos algún estilo para ver mejor cada sección pero recuerda que estamos con HTML5. Tienes que aprender a identificar componentes y codificarlos bien.



# Estructura del HTML

- Tarea: Realiza el siguiente formato de formulario en HTML5

1. Name

2. Age

3. Url

4. Email

5. Phone

6. Address

7. Post code

8. Country

9. Gender  
 Male  female

NO hay que hacer nada de estilos CSS, sólo la **estructura**. Para los iconos puedes buscarlos en FontAwesome o usar imágenes que tengas que adaptar en tamaño. Coloca bien los label para cada input y asocialos con for y name respectivamente.

**Step 1: Your details**

Name

Email

Phone

**Step 2: Delivery address**

Address

Post code

Country

**Step 3: Card details**

Card type  
 VISA    AmEx    Mastercard

Card number

Security code

Name on card



# Estructura del HTML

- **Elementos Multimedia en HTML:** Usaremos: <https://myfreemp3.to/> y Youtube

- Ya se ha visto cómo incorporar imágenes y enlaces a ficheros a nuestra página web. Con HTML5 también se permite incorporar sonido y vídeo o incluso pintar en la propia página web. Esto es lo que se conoce como elementos multimedia.
- Sonido: HTML5 ha proporcionado un estándar para escuchar los ficheros de audio ya que en versiones anteriores era necesario un plug-in. Es importante ofrecer distintas alternativas de formato . Se inserta en HTML mediante la etiqueta <audio>.

```
<audio controls>
  <source src="cancion.ogg" type="audio/ogg">
  <source src="cancion.mp3" type="audio/mpeg">
```

Your browser does not support the audio element. Si no pongo control no saldrían los controles de reproducción. También puedo poner después autoplay y mute



- Vídeo: El video en HTML se puede insertar con la etiqueta <video> o <iframe> o <embed> para vídeos de youtube. En el caso de <video> es importante ofrecer distintas alternativas de formato al igual que ocurría para el audio. Hay atributos interesantes que puedes ver aquí

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

```
<iframe width="420" height="315"
src="http://www.youtube.com/nombreVideo">
</iframe>
```

**Práctica:** ¿Y si quiero montar una [playlist](#)?  
Tal y como lo vemos en páginas requiere JavaScript



# Estructura del HTML

- **Elementos Multimedia:** en HTML5

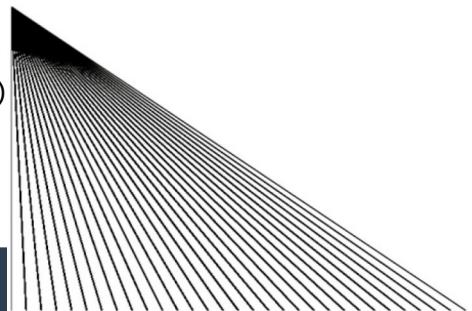
- Si queremos, podemos incluir subtítulos a un vídeo con la etiqueta `track` donde se indica el origen o el idioma que haya en el fichero `vtt` con el siguiente formato:

```
<track src="spa.vtt" kind="subtitles" srclang="es" label="Español" >
```

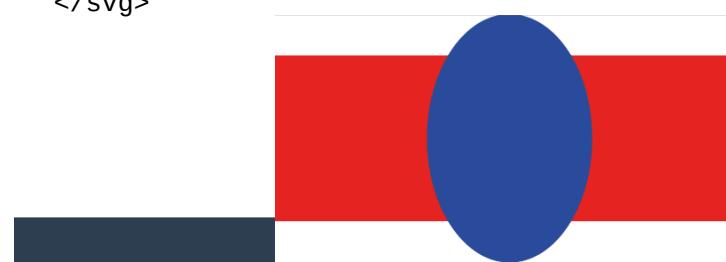
```
WEBVTT
1 dummyfile.vtt
2
3 0
4 00:07.410 --> 00:11.660 Timeframe
5 Welcome everyone! This is an example of a vtt file.
6
7 1
8 00:11.670 --> 00:16.070 Content
9 This helps to add subtitles or captions for a video file.
10
11 2
12 00:16.080 --> 00:16.070
13 So, this is the end of this file.
```

- También podemos dibujar `<canvas>` o colocar imágenes SVG o ecuaciones `<math>`. Su contenido indica el formato de lo que van a mostrar. Aquí hay unos ejemplos:

```
<canvas id="lienzo1" width="600" height="400">
    <script type="text/javascript">
        var canvas=document.getElementById("lienzo1");
        var contexto=canvas.getContext("2d");
        contexto.lineWidth=2;
        for(i=0;i<=600;i+=20){
            contexto.moveTo(0,0);
            contexto.lineTo(i,400)
            contexto.stroke();
        }
    </script>
</canvas>
```



```
<svg xmlns="http://www.w3c.org/2000/svg" height="400">
    <rect id="rec1" x="50" y="50" width="300"
          height="100" fill="red" />
    <ellipse id="elips1" cx="200" cy="100" rx="50"
             ry="75" fill="blue" />
</svg>
```





# Estructura del HTML

- **Otros elementos en HTML5**
  - **Iframe**: <https://developer.mozilla.org/es/docs/Web/HTML/Element/iframe>
  - **Embed**: <https://developer.mozilla.org/es/docs/Web/HTML/Element/embed>
  - **Span**: <https://developer.mozilla.org/es/docs/Web/HTML/Element/span>
  - **Object**: <https://developer.mozilla.org/es/docs/Web/HTML/Element/object>
- **Accesibilidad en HTML5**: Por ley, sobre todo para trabajar con la administración, tenemos que preparar nuestra web para que sea accesible, especialmente para personas con problemas visuales. ¿Cómo gestionamos esto?
  - Utilizar etiquetas semánticas de HTML5 en lugar de usar bloques div
  - Utilizar for en input para indicar qué hay ahí
  - Especificar en alt el contenido de un vídeo o una imagen
  - Especificar el idioma de la página y el juego de caracteres
  - Colocar atributos ARIA (Accessible Rich Internet Application): <https://accesible.es/wai-aria> y <https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Attributes/aria-live> ([ejemplo](#))

**Importante:** Se ha hecho un uso masivo de div para estructurar la web por “cajas” pero si hablamos de contenido, utiliza las etiquetas section y article ([enlace](#))

Hay bastante lío en el uso de section y article y ante la duda se usará div pero con criterio



# Estructura del HTML

- **Práctica:** Con lo que ya sabemos, vamos a intentar copiar la estructura de la siguientes webs. Usa las etiquetas HTML5 (**header**, **nav**, **main**, **section**, **article**, **aside**, **footer**)  
**¡Importante!** No todas las páginas tienen que tener TODO o que haya más de uno como n los articles. Para los iconos de imagen usa esta [web](#)

The screenshot shows a Moodle course interface with the following details:

- Header:** moodle CENTROS CURSO 22/23, Centros: I.E.S. Cura Valera, Junta de Andalucía logo.
- Navigation:** A vertical sidebar on the left contains icons for Home, Courses, Groups, Books, and Folders, with the Folders icon highlighted.
- Main Content Area:** Three course modules are displayed:
  - Fundamentos de hardware:** Shows a collage of various computer hardware components like monitors, servers, and printers. Below it are buttons for CUESTIONARIO, TAREAS, FOROS, CALIFICACIÓN, and RECURSOS.
  - Gestión de bases de datos:** Displays a database schema diagram with entities Customer, Order, Product, and Invoice, and their relationships. Below it are buttons for CUESTIONARIO, TAREAS, FOROS, CALIFICACIÓN, and RECURSOS.
  - Lenguajes de marcas y sistemas de gestión de ...:** Shows a large red 3D logo. Below it are buttons for CUESTIONARIO, TAREAS, FOROS, CALIFICACIÓN, and RECURSOS.
- Footer:** Buttons for Mostrar 99, Back, and Forward.



# Estructura del HTML

- **Actividad:** Con lo que ya sabemos, vamos a intentar copiar la estructura de la siguientes webs. Usa las etiquetas HTML5. Para los iconos de imagen usa esta web: <https://fonts.google.com/icons> o <https://fontawesome.com/>

The screenshot shows the homepage of the IES CURA VALERA website. At the top, there is a banner with a photograph of the school building and courtyard. On the left of the banner is the school's logo and name. On the right is a search bar. Below the banner is a navigation menu with links for 'Inicio', 'Noticias', 'Oferta Educativa', 'Vídeos y Fotos', and 'Contacto'. The main content area features a large photograph of the school's exterior and courtyard. At the bottom, there are three cards with links for 'ADULTOS...', 'Horarios Nocturno 2022/2023', and 'Reunión para comienzo de curso'. In the bottom right corner, there is a logo for 'BIBLIOTECA WEB biblioweb'.

ADULTOS...

Acceso a la plataforma de

ADULTOS...

Horarios Nocturno 2022/2023

ADULTOS

Reunión para comienzo de curso

BIBLIOTECA WEB

biblioweb



# Estructura del HTML

- **Tarea:** Desarrolla en HTML5 un portal web similar a XATAKA

**Estructura de capas:** Vamos a crear bloques div para organizar luego en CSS su posición

Dentro de <body> incluye las etiquetas <header>, <nav>, <main> y <footer>.

En <main> se incluirán las capas <section> y <aside>.

En <aside> se dividirá en <div id="recomendados"> y <div id="escribenos">.

En <footer> se dividirá en cinco bloques div, uno para cada lista desordenada.

Incluye el ícono en la web (puede ser de la web o uno creado por tí) y coloca las etiquetas meta (informática, hardware) en <head>

En <header>: Incluye una imagen alargada. En este caso XATAKA

En <nav>: El menú incluirá una lista no ordenada con enlaces a la propia web del instituto. Tendrá: Acceso al centro, Un submenú: Educación [Educación General, Ciclos Formativos, adultos], Biblioweb, Noticias, Contacto y Login.

En <main>: Crearás una sección que incluirá dos artículos con noticias relevantes de tecnología. Cada una tendrá un texto formateado e imágenes pero la primera noticia va a incorporar la tabla de la web <https://www.comprarhosting.co/comparacion-ryzen-3-5-7/> y la segunda añadirá un vídeo de Youtube sobre cableado estructurado.

Los iconos de "Síguenos" serán una sección de redes sociales con iconos de fontawesome individuales, y cada uno será un enlace a la red social correspondiente.

En <aside>: Será un bloque div que tendrá 3 noticias compuestas por una imagen y un texto descriptivo. La imagen llevará a la noticia real. Después incorpora en otro div un formulario con la estructura de la imagen.

En <footer>: Se crearán 5 div, uno por cada apartado que hay en el footer de <https://www.xataka.com/> con el mismo contenido. Después, se pondrá un último enlace con ícono de inicio para volver hacia el menú utilizando un ancla.

No te preocupes por el aspecto visual, una cosa es CÓMO queremos que se vea y otra QUÉ queremos que tenga. Ahora nos estamos preocupando por dar estructura y contenido y respondemos al QUÉ

The screenshot shows the Xataka website's layout. At the top is a header with the Xataka logo and navigation links. Below it is a main content area with a large image and text about keyboard shortcuts. Further down are sections for recommended news and a contact form. The footer contains links to various Xataka sub-sites and social media icons.

# Actividades de refuerzo y ampliación

| **Tienes que entender que el desarrollo web tiene mucho componente artístico y que requiere mucho tiempo para el diseño que es lo que veremos en el tema siguiente con CSS. ¡No te apresures a poner estilos dentro del HTML!**

| **Actividades de refuerzo:** Realiza la relación de ejercicios de la unidad  
| **Actividad de ampliación:** Haz un glosario con las etiquetas que han dejado de utilizarse tras aparecer HTML5 o intenta replicar la web de MediaMarkt