



**Talento
Digital
para Chile:**

**Módulo 1
Fundamentos del
Desarrollo Web**



MÓDULO 1 – FUNDAMENTOS DEL DESARROLLO WEB

1.1.- Contenido 1: Estructurar contenido en una página web utilizando HTML5 para dar solución a una problemática

Objetivo de la jornada

1. Identifica las etiquetas básicas para agregar contenido a una página de acuerdo al lenguaje html para dar solución a una problemática.
2. Estructura una página web entorno a sus componentes clave utilizando un editor de código para la codificación y el inspector de elementos del navegador para identificar elementos de la página.
3. Estructura assets del proyecto, ya sean, imágenes o archivos css, utilizando rutas locales, absolutas o CDN de acuerdo a las buenas prácticas de la industria.

1.1.1.- El Lenguaje HTML

1.1.1.1. Internet, la World Wide Web y el protocolo HTTP

INTERNET

Desde la construcción del primer computador programable de la historia - El Manchester Baby en 1948 en la Universidad de Manchester del Reino Unido - y hasta los años 60s, la programación de computadoras fue una tarea enfocada en secuencias de instrucciones lógico/matemáticas ejecutadas localmente en máquinas aisladas unas de otras, que entregaban sus resultados a través de alguna interfaz con el usuario, tales como pantallas o impresoras^[1].

El paradigma descrito cambió con la aparición de nuevas ideas en los años 60, como la "Red Galáctica" de J.C.R. Licklider del MIT en 1962 en la que imaginaba una red de computadores conectados globalmente a través de los que todo el mundo podría acceder rápidamente a datos y programas desde cualquier sitio. Licklider era director del programa de informática de DARPA (Agencia de Proyectos de Investigación Avanzada de Defensa de Estados Unidos).



La primera prueba de interacción remota de computadores fue realizada por Lawrence G. Roberts, investigador del MIT, quien en 1965 conectó exitosamente la máquina TX-2 desde Massachusetts, con el Q-32, en California, mediante una línea telefónica conmutada de baja velocidad, logrando hacerlos trabajar en tiempo compartido intercambiando programas y datos. Sin embargo, la red telefónica no era la plataforma adecuada y se vio la necesidad de pensar en las ideas ya existentes de comunicación por paquetes conmutados de datos planteadas por Leonard Kleinrock, del MIT ya en 1961.

Roberts ingresó en a DARPA en 1966, desarrollando su plan de redes informáticas y creando un plan llamado ARPANET, que publicó en 1967. Paralelamente en el Reino Unido se habían desarrollado ideas similares, por Donald Davies, Roger Scantlebury y Paul Baran. Distintas instituciones habían trabajado simultáneamente en ideas similares sin saberlo: MIT (1961-1967), de RAND (1962-1965) y del NPL (1964-1967). Luego de intercambio de ideas entre estos grupos se consolidó el concepto de “Paquete” de datos y la velocidad de ARPANET se definió en 50kbps. Con esto, se solicitó presupuesto para su construcción y de esta manera, a finales de 1969, había cuatro hosts conectados en la ARPANET inicial: Network Measurement Center de UCLA, Stanford Research Institute (SRI), el nodo de la Universidad de California en Santa Bárbara y el de la Universidad de Utah. Así Internet daba inicio a su trayectoria.

En 1972 se hizo la primera demostración pública de este tipo de redes. Además también se introdujo la aplicación “hot” inicial, el correo electrónico. En marzo, Ray Tomlinson, de BBN, escribió el software básico de envío y lectura de mensajes de correo electrónico, motivado por la necesidad de los desarrolladores de ARPANET de un mecanismo sencillo de coordinación. En julio, Roberts amplió su utilidad escribiendo la primera utilidad de correo electrónico para hacer listas de mensajes, leerlos selectivamente, archivarlos, reenviarlos y responder a los mismos. A partir de ese momento, el correo electrónico se convirtió en la aplicación de red más importante durante más de una década. Esto presagió el tipo de actividad que vemos hoy en día en la World Wide Web.

Una clave para el rápido crecimiento de Internet ha sido el acceso abierto y gratuito a los documentos básicos, en especial las especificaciones de los protocolos.

Los principios de ARPANET e Internet en la comunidad investigadora universitaria promovieron la tradición académica de publicar ideas y resultados de forma abierta. Sin embargo, el ciclo normal de las publicaciones académicas tradicionales era



demasiado formal y demasiado lento para el intercambio dinámico de ideas esencial para la creación de redes.

En 1969, S. Crocker (en UCLA en aquel momento) dio un paso esencial al establecer las series de notas Petición de comentarios (o RFC). La idea de estos memorandos era que fuesen una forma de distribución informal y rápida para compartir ideas con otros investigadores de la red. Al principio, las RFC se imprimían en papel y se distribuían a través del correo ordinario. Cuando se empezó a usar el Protocolo de Transferencia de Archivos (FTP), las RFC se preparaban como archivos en línea y se accedía a ellas a través de FTP. Hoy en día, por supuesto, se accede a las RFC fácilmente a través de la World Wide Web en numerosos sitios de todo el mundo. El SRI, en su papel de Centro de Información de Redes, mantenía los directorios en línea. Jon Postel fue editor de RFC además de gestionar la administración centralizada de las asignaciones necesarias de números de protocolo, trabajos que ejerció hasta su muerte, el 16 de octubre de 1998.

El efecto de las RFC fue crear un bucle de comentarios positivos, en el que las ideas y propuestas presentadas en una RFC desencadenaban otra RFC con ideas adicionales, y así sucesivamente. Cuando se conseguía un consenso (o al menos un conjunto coherente de ideas), se preparaba un documento de especificaciones. Después, varios equipos de investigación usaban esas especificaciones como base para las implementaciones.

Con el tiempo, las RFC se han ido centrando más en los estándares de los protocolos (las especificaciones “oficiales”), aunque siguen existiendo RFC informativas que describen enfoques alternativos, u ofrecen información sobre los antecedentes de los protocolos y los problemas de ingeniería. Hoy en día, las RFC se conciben como “documentos oficiales” en la comunidad de ingeniería y estándares de Internet.

El acceso abierto a las RFC (gratuito, si tiene cualquier tipo de conexión a Internet) promueve el crecimiento de Internet porque permite usar las especificaciones reales como ejemplos en clases y entre los emprendedores que desarrollan nuevos sistemas.

El correo electrónico ha sido un factor significativo en todas las áreas de Internet, y eso es especialmente cierto en el desarrollo de especificaciones de protocolos y estándares técnicos y en la ingeniería de Internet. Las primeras RFC solían presentar un conjunto de ideas desarrolladas por un determinado grupo de investigadores, ubicado en un punto concreto, que las presentaban al resto de la comunidad.



Cuando se empezó a usar el correo electrónico, el patrón de autoría cambió: las RFC eran presentadas por varios autores con una visión común, independientemente de su ubicación.

El uso de listas de correo electrónico especializadas se ha usado desde hace tiempo en el desarrollo de especificaciones de protocolos, y sigue siendo una herramienta importante. Ahora la IETF consta de más de 75 grupos de trabajo, cada uno trabajando en un aspecto diferente de la ingeniería de Internet. Cada uno de esos grupos de trabajo tiene una lista de correo electrónico para discutir uno o más borradores en vías de desarrollo. Cuando se alcanza el consenso sobre un borrador, se puede distribuir como RFC (<https://www.ietf.org/standards/>).

Como la actual y rápida expansión de Internet está impulsada por la conciencia de su capacidad para compartir información, deberíamos entender que el primer papel de la red a la hora de compartir información fue compartir la información relativa a su propio diseño y funcionamiento a través de las RFC. Este método único de desarrollar nuevas funciones en la red seguirá teniendo una importancia fundamental en la evolución futura de Internet^[2].

WORLD WIDE WEB

En los años 80, el ingeniero inglés Tim Berners-Lee, trabajando en el CERN, el laboratorio de física de partículas de Suiza, detectó la problemática que enfrentaban los científicos para compartir información entre ellos. Berners-Lee visualizó la posibilidad de explotar para estos fines una tecnología emergente llamada Hipertexto. Tim emitió un documento que denominó **Manejo de la Información: Una Propuesta**, que es la base de lo que conocemos como Web, el cual su jefe catalogó como interesante pero vago. La Web nunca fue un proyecto formal de CERN pero se le permitió a Tim trabajar en éste. En octubre de 1990, Berners-Lee había generado las tres tecnologías fundamentales que pasaron a ser los fundamentos de lo que conocemos actualmente como Web^[3]:

- **HTML (HyperText Markup Language):** El lenguaje de marcas para la Web.
- **URI (Uniform Resource Identifier):** Un cierto tipo de “dirección”, única y utilizada para identificar a cada recurso en la Web. Ésta es llamada comúnmente URL.
- **HTTP (Hypertext Transfer Protocol):** Protocolo de comunicaciones que permite el intercambio de recursos enlazados a través de la Web.



Tim también creó el primer editor/navegador de páginas web ("WorldWideWeb.app") y el primer servidor web ("httpd"). Para fines de 1990, la primera página web estaba "al aire" en la internet abierta y desde 1991 se invitó a nuevos participantes externos al CERN a formar parte de esta comunidad Web. Se promovió una filosofía abierta para esta nueva tecnología, y Berners-Lee manifestó que "De tratarse de una tecnología propietaria, y bajo mi total control, ésta probablemente no habría despegado. No puede proponerse algo con la intención de hacerlo universal y al mismo tiempo mantener control sobre esto".

El CERN aceptó liberar los códigos relacionados con estas tecnologías para siempre. Esta decisión se anunció en abril de 1993, y detonó una ola global de creatividad, colaboración e innovación nunca antes vista. En 2003, las compañías que desarrollaban nuevas tecnologías Web, se comprometieron a una política de apertura en sus trabajos.

Tim se migró desde el CERN al MIT en 1994 para fundar el World Wide Web Consortium (W3C), una comunidad internacional dedicada al desarrollo de estándares abiertos para la Web. Él se mantiene como director del W3C hasta la actualidad.

La comunidad web temprana produjo algunas ideas revolucionarias que actualmente se extienden más allá del sector tecnológico:

- **Descentralización:** Ausencia de una autoridad central que entregue permiso para publicar cualquier contenido en la Web. No hay nodo controlador central. No hay un punto de falla único, ni un "switch de apagado", lo que implica libertad frente a censura y vigilancia indiscriminada.
- **No-Discriminación:** Si yo pago por conectarme a la Internet con cierta calidad de servicio, y tú pagas para conectarte con la misma calidad de servicio o superior, entonces ambos podemos comunicarnos al mismo nivel. Este principio de igualdad es también conocido como Neutralidad de la Red.
- **Diseño Bottom-Up:** En lugar de código escrito y controlado por un pequeño grupo de expertos, se opta por un desarrollo abierto a todos, invitando a una máxima participación y experimentación.
- **Universalidad:** Para que cualquier persona pueda publicar cualquier cosa en la Web, todos los computadores involucrados deben hablar los mismos lenguajes unos a otros, sin importar qué tipo de hardware estén utilizando; dónde vivan; o qué creencias políticas o culturales tengan.



- **Consenso:** Para que los estándares universales funcionen, todos deben acordar usarlos. Tim y otros lograron este consenso entregando a todos una voz en la creación de los estándares, a través de un proceso transparente y participatorio en el W3C.

HTTP (HYPERTEXT TRANSFER PROTOCOL)

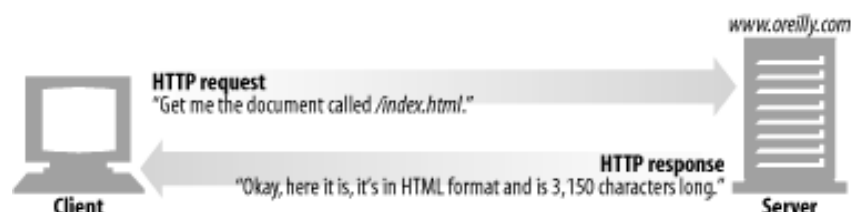
HTTP es el protocolo que utilizan los programas para comunicarse en le World Wide Web. Hay muchas aplicaciones de HTTP, pero es principalmente conocido como la herramienta de conversación entre Navegadores y Servidores Web. HTTP es el lenguaje de la Internet Global^[5].

Miles de millones de imágenes JPEG, páginas HTML, archivos de texto, películas MPEG, archivos de audio WAV y otros contenidos navegan por Internet todos los días. HTTP mueve la mayor parte de esta información de manera rápida, conveniente y confiable desde servidores web de todo el mundo a navegadores web en los escritorios de personas.

Debido a que HTTP utiliza protocolos confiables de transmisión de datos, garantiza que sus datos no se dañarán ni se mezclarán en tránsito, incluso cuando provengan del otro lado del mundo. Esto es bueno para nosotros como usuarios, ya que podemos acceder a la información sin preocuparnos por su integridad. Una transmisión confiable también es buena para nosotros como desarrolladores de aplicaciones de Internet, ya que no necesitamos preocuparnos de que las comunicaciones HTTP sean destruidas, duplicadas o distorsionadas en el tránsito. Podemos concentrarnos en programar los detalles distintivos de nuestra aplicación, sin preocuparnos por los defectos y las debilidades de Internet.

CLIENTES Y SERVIDORES WEB

El contenido web reside en servidores web. Los servidores web hablan el protocolo HTTP, por lo que a menudo se les llama Servidores HTTP. Estos servidores HTTP almacenan los datos de Internet y proporcionan los datos cuando son solicitados por clientes HTTP. Los clientes envían solicitudes HTTP a los servidores, y los servidores devuelven los datos solicitados en Respuestas HTTP, tal como muestra la siguiente figura.



Requerimiento y Respuesta entre Cliente y Servidor HTTP

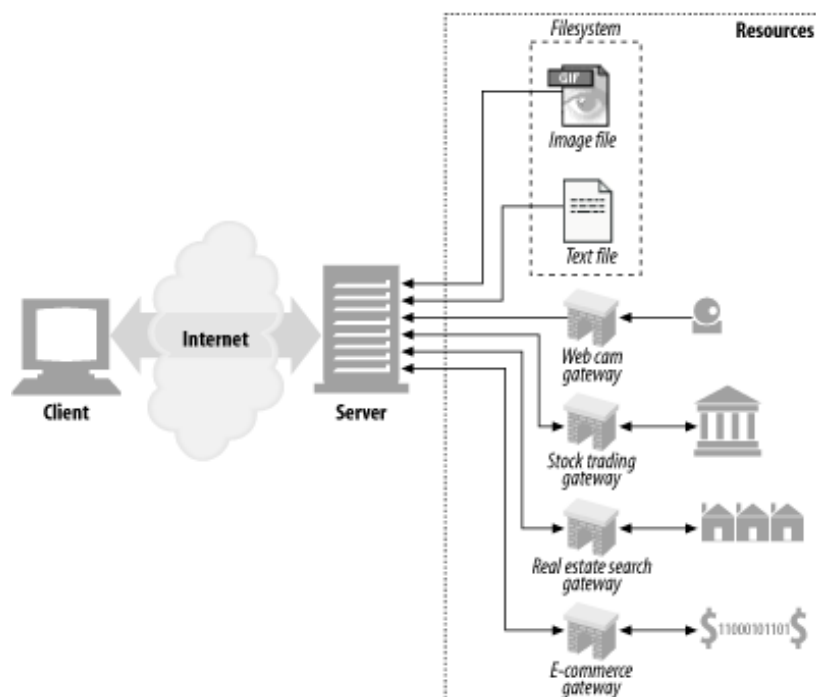
Los clientes HTTP y los servidores HTTP son los componentes básicos de la World Wide Web.

Otro término que es ampliamente utilizado en este contexto es **Agente de Usuario (User Agent)** se usa para denominar cualquier software que obtiene y presenta contenido Web para usuarios finales, o es implementado usando tecnologías Web. Éstos incluyen Navegadores Web, Media Players, y plug-ins que ayudan a obtener, desplegar e interactuar con contenido Web. La familia de Agentes de Usuario también incluye shells de sistemas operativos, electrodomésticos con web-widgets, aplicaciones stand-alone o aplicaciones electrónicas cuyas interfaces de usuario estén implementadas como una combinación de tecnologías Web^[4].

RECURSOS WEB

Los servidores web contienen **Recursos Web**. Un recurso web es la fuente del contenido web. El tipo más simple de recurso web es un archivo estático en el sistema de archivos del servidor web. Estos archivos pueden contener cualquier cosa: pueden ser archivos de texto, archivos HTML, archivos de Microsoft Word, archivos PDF, archivos de imagen JPEG, archivos de película MP4 o cualquier otro formato.

Sin embargo, los recursos no tienen que ser archivos estáticos. Los recursos también pueden ser programas de software que generan contenido a pedido. Estos recursos de contenido dinámico pueden generar contenido en función de la identidad del usuario del Cliente HTTP, de la información que solicitó o de la hora del día. Pueden mostrarle una imagen en vivo desde una cámara o permitirle transar acciones, buscar en bases de datos de bienes raíces o comprar regalos en tiendas en línea.



Recursos Web provistos por un HTTP Server

En resumen, un recurso es cualquier tipo de fuente de contenido. Un archivo que contiene la planilla de cálculo del pronóstico de ventas de una empresa. Un portal web para realizar búsquedas en los estantes de una biblioteca es un recurso. De la misma forma, un motor de búsqueda de Internet es un recurso.

MEDIA TYPES

Debido a que Internet aloja muchos miles de tipos de datos diferentes, HTTP etiqueta cuidadosamente cada objeto que se transporta a través de la Web con una etiqueta de formato de datos llamada tipo MIME. MIME (Extensiones multipropósito de correo de Internet) fue originalmente diseñado para resolver problemas encontrados al mover mensajes entre diferentes sistemas de correo electrónico. MIME funcionó tan bien para el correo electrónico que HTTP lo adoptó para describir y etiquetar su propio contenido multimedia.

Los servidores web adjuntan un tipo MIME a todos los datos de objetos HTTP. Cuando un navegador web recupera un objeto de un servidor, mira el tipo MIME asociado para ver si sabe cómo manejar el objeto. La mayoría de los navegadores pueden manejar cientos de tipos de objetos populares: mostrar archivos de imagen, analizar y formatear archivos HTML, reproducir archivos de audio a través de los altavoces de la computadora o iniciar un software de complemento externo para manejar formatos especiales.



Ejemplos de Media Types, entre los cientos existentes, son:

- Documento de texto en formato HTML: **text/html**.
- Documento de texto plano ASCII: **text/plain**.
- Una versión JPEG de una imagen: **image/jpeg**.
- Una versión GIF de una imagen: **image/gif**.

URIs

Cada recurso de servidor web tiene un nombre, por lo que los clientes pueden señalar qué recursos les interesan. El nombre del recurso del servidor se denomina identificador uniforme de recursos o **URI**. Los URI son similares las direcciones postales pero en Internet, identificando y localizando recursos de información en todo el mundo.

Por ejemplo, una URI para un recurso de imagen en el servidor awakelab.com puede ser:

<http://www.servidordeesteejemplo.com/ejemplos/meme.gif>

Las URIs vienen en dos tipos, llamados URL (Uniform Resources Locator) and URN (Uniform Resource Name).

URLs

La URL es la forma más común de identificador de recursos. Las URLs describen la ubicación específica de un recurso en un servidor en particular. Dicen exactamente cómo obtener un recurso desde una ubicación precisa y fija, por ejemplo:

<http://www.servidordeesteejemplo.com/specials/saw-blade.gif>

La mayoría de las URL siguen un formato estandarizado de tres partes principales:

- La primera parte de la URL se llama esquema y describe el protocolo utilizado para acceder al recurso. Este suele ser el protocolo HTTP (<http://>), pero puede ser otro protocolo como FTP (<ftp://>).



- La segunda parte proporciona la dirección de Internet del servidor (por ejemplo, www.servidordeesteejemplo.com).
- El resto de la URL nombra un recurso en el servidor web (por ejemplo, `/specials/saw-blade.gif`).

En la actualidad, casi cada URI es una URL.

URNs

El segundo tipo de URI es la URN. Una URN sirve como un nombre único para un contenido en particular, independientemente del lugar donde reside actualmente el recurso. Estas URN independientes de la ubicación permiten que los recursos se muevan de un lugar a otro. Los URN también permiten que los protocolos de acceso a la red accedan a los recursos manteniendo el mismo nombre.

Por ejemplo, la siguiente URN podría usarse para nombrar el documento de estándares de Internet "RFC 2141" independientemente de dónde resida (incluso puede copiarse en varios lugares):

[urna:ietf:rfc:2141](urn:ietf:rfc:2141)

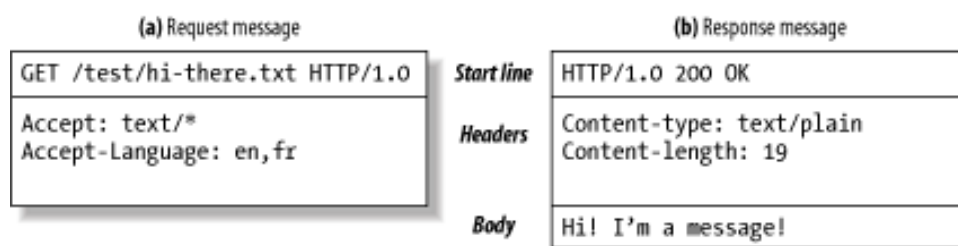
Para trabajar de manera efectiva, las URN necesitan una infraestructura de soporte para resolver ubicaciones de recursos.

A menos que se indique lo contrario, generalmente se adoptan de manera intercambiable los términos URI y URL.

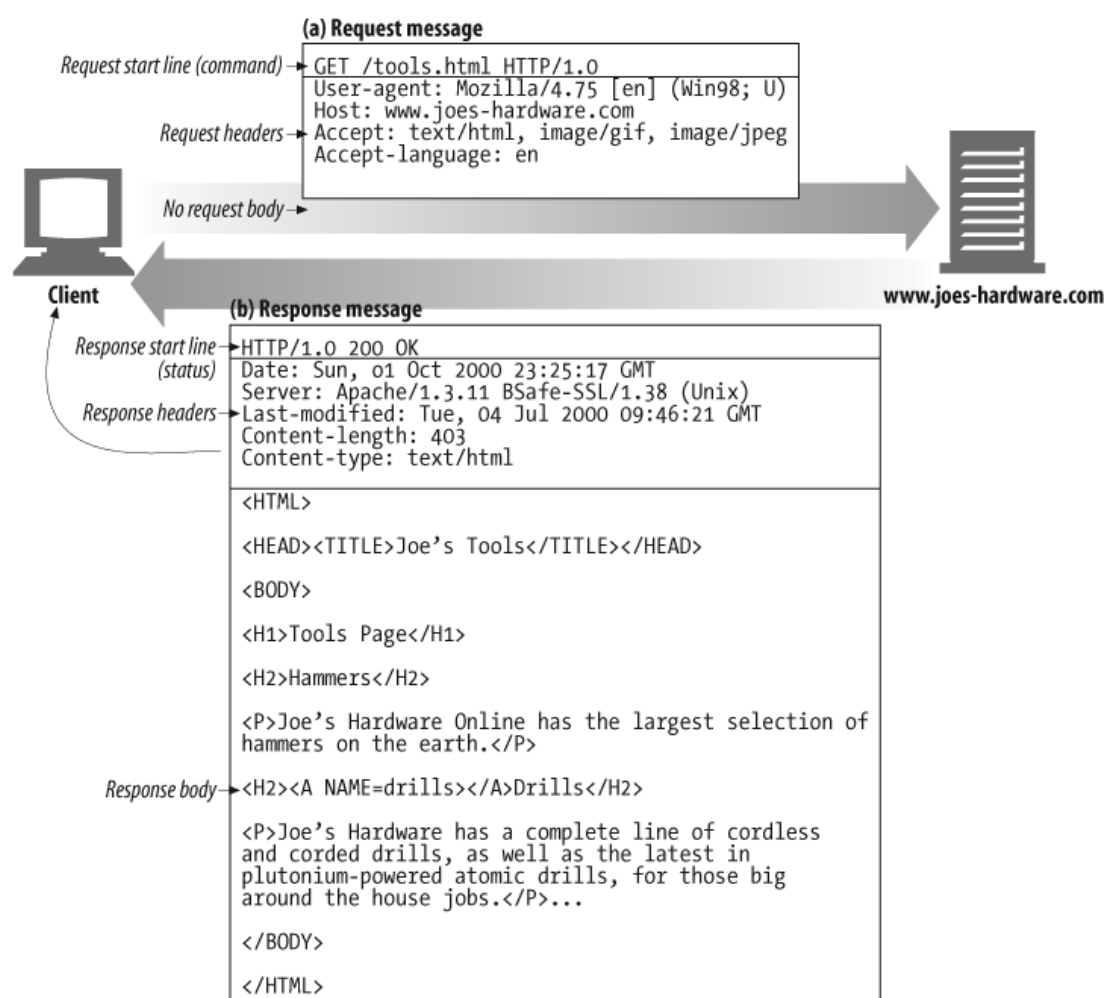
MENSAJES

Los mensajes HTTP son secuencias de caracteres simples, orientadas a líneas. Debido a que son texto sin formato, no binario, son fáciles de leer y escribir para los humanos.

El mensaje HTTP enviado desde clientes web a servidores web se denomina **Request**. El mensaje de los servidores a los clientes se denomina **Response**. No hay otros tipos de mensajes HTTP. Los formatos de los mensajes HTTP de request y reponse son muy similares y siguen la estructura de Start Line, Headers y Body, como se muestra a continuación:



Un ejemplo más concreto de una transacción Request/Response entre un cliente y un servidor HTTP es:



Cada mensaje de respuesta HTTP es emitido con un código de estado. El código de estado es un código numérico de tres dígitos que le dice al cliente si la solicitud se realizó correctamente o si se requieren otras acciones. Algunos códigos de estado comunes son:



Status Code HTTP	Descripción
200	OK. Request exitoso.
302	Redirect. El recurso de la URI solicitada ha sido cambiado temporalmente.
404	Not Found. El servidor no pudo encontrar el contenido solicitado.

Más códigos en: <https://developer.mozilla.org/es/docs/Web/HTTP/Status>

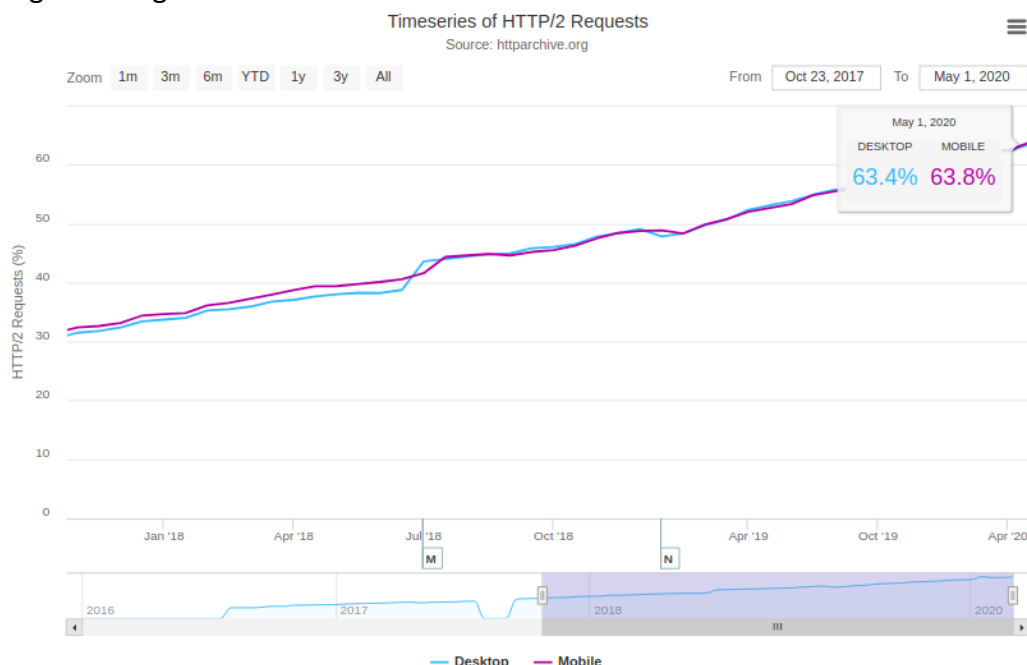
Las versiones de protocolo HTTP utilizada en los ejemplos anteriores son a modo ilustrativo. La versión de protocolo HTTP 1.1 (<https://tools.ietf.org/html/rfc2616>) ha sido utilizada por un tiempo extenso, y aún es muy utilizada en la actualidad. Las versiones existentes del protocolo HTTP a través de los años se muestra en la siguiente figura:



Versiones protocolo HTTP en el tiempo

(Fuente: https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto)

La adopción de protocolos de forma masiva depende de muchos factores y no necesariamente ocurre de forma inmediata luego de publicado el estandar. En caso del protocolo HTTP 2, su estado de adopción en la actualidad es como se muestra en la siguiente figura:



Estado de adopción protocolo HTTP2

(Fuente: <https://httparchive.org>)



LENGUAJES LADO CLIENTE

Un lenguaje de lado cliente es aquel que se procesa dentro del navegador del cliente. Un código de lado cliente generalmente de un tamaño razonable para que el navegador lo descargue en un tiempo razonable y que no entorpezca la experiencia del usuario. JavaScript es un lenguaje de programación popular de lado del cliente y ampliamente utilizado en sitios web dinámicos. El programa se puede incrustar en el código HTML o ser almacenado en un archivo externo. Su ejecución se produce al completar la carga de la página en el navegador o como resultado de alguna acción del usuario, como presionar un botón de la página. HTML y CSS también pertenecen a los lenguajes del lado cliente, tal vez no como lenguajes de programación propiamente tales, pero sí como lenguajes de marcado y estilo.

LENGUAJES LADO SERVIDOR

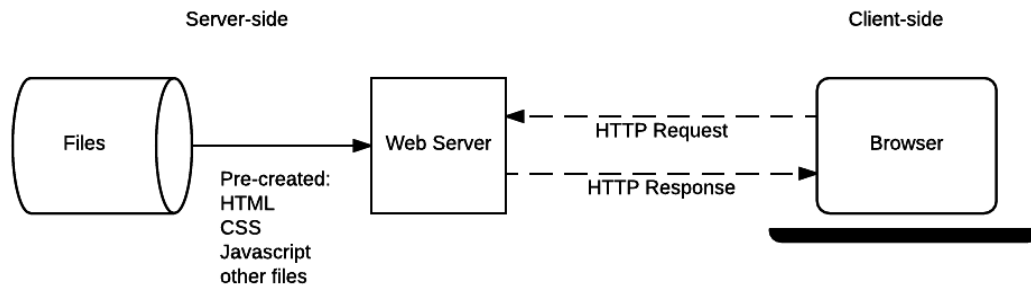
Un código en lenguaje de lado servidor se procesa, como dice su nombre, en el servidor web cuando el usuario solicita información. Este tipo de programa puede ejecutarse antes de cargar una página web. Son necesarios para cualquier cosa que requiera datos dinámicos, como almacenar los detalles de inicio de sesión del usuario. Algunos lenguajes comunes del lado del servidor incluyen PHP, Python, Ruby y Java. Estos se ejecutan como lenguajes de programación en el servidor.

Cuando se procesa un programa de lado servidor, la solicitud se envía a éste y el resultado se devuelve al cliente. Esto es útil para sitios web que almacenan grandes cantidades de datos, como motores de búsqueda o redes sociales; sería muy lento para el navegador del cliente descargar todos los datos.

SITIOS WEB ESTÁTICOS

El siguiente diagrama muestra una arquitectura básica de servidor web para un sitio estático (Un sitio estático es aquel que devuelve el mismo contenido codificado del servidor cada vez que se solicita un recurso en particular). Cuando un usuario quiere navegar a una página, el navegador envía una solicitud HTTP "GET" especificando su URL^[6].

El servidor recupera el documento solicitado de su sistema de archivos y devuelve una respuesta HTTP que contiene el documento y un estado de éxito (generalmente 200 OK). Si el archivo no se puede recuperar por algún motivo, se devuelve un estado de error (consulte las respuestas de error del cliente y las respuestas de error del servidor).



SITIOS WEB DINÁMICOS

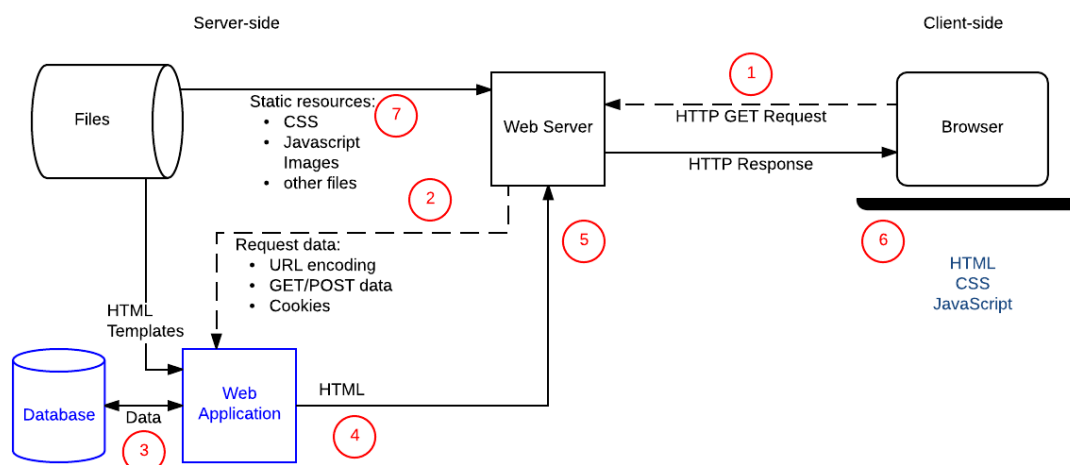
Un sitio web dinámico es aquel en el que parte del contenido de respuesta se genera dinámicamente, solo cuando es necesario. En un sitio web dinámico, las páginas HTML normalmente se crean insertando datos de una base de datos en marcadores de posición en plantillas HTML (Esta es una forma mucho más eficiente de almacenar grandes cantidades de contenido que usar sitios web estáticos).

Un sitio dinámico puede devolver diferentes datos para una URL en función de la información proporcionada por el usuario o las preferencias almacenadas y puede realizar otras operaciones como parte de la devolución de una respuesta (por ejemplo, envío de notificaciones).

La mayor parte del código para dar funcionamiento a un sitio web dinámico debe ejecutarse en el servidor. La creación de este código se conoce como "programación de lado servidor".

El siguiente diagrama muestra una arquitectura simple para un sitio web dinámico. Como en el diagrama anterior, los navegadores envían solicitudes HTTP al servidor, luego el servidor procesa las solicitudes y devuelve las respuestas HTTP apropiadas.

Las solicitudes de recursos estáticos se manejan de la misma manera que para los sitios estáticos (los recursos estáticos son archivos que no cambian, típicamente: CSS, JavaScript, imágenes, archivos PDF creados previamente, etc.).



1.1.1.2. HTML

A continuación se extiende más en detalle lo introducido en la sección anterior sobre HTML (HyperText Markup Language), el lenguaje de marcas central de la World Wide Web. Originalmente, HTML fue diseñado principalmente como un lenguaje para describir semánticamente documentos científicos. Sin embargo, su diseño general le ha permitido adaptarse, en los años siguientes, para describir otros tipos de documentos e incluso aplicaciones.

Durante sus primeros cinco años (1990-1995), HTML pasó por varias revisiones y experimentó una serie de extensiones, principalmente alojadas primero en el CERN y luego en el IETF.

Con la creación del W3C, el desarrollo de HTML cambió de lugar nuevamente. Un primer intento fallido de extender HTML en 1995 conocido como HTML 3.0 luego dio paso a un enfoque más pragmático conocido como HTML 3.2, que se completó en 1997. HTML4 siguió rápidamente más tarde ese mismo año.

Al año siguiente, la membresía del W3C decidió dejar de evolucionar HTML y, en cambio, comenzó a trabajar en un equivalente basado en XML, llamado XHTML. Este esfuerzo comenzó con una reformulación de HTML4 en XML, conocido como XHTML 1.0, que no agregó nuevas características, excepto la nueva serialización, y que se completó en 2000. Después de XHTML 1.0, el enfoque del W3C se volvió más fácil para otros grupos de trabajo. extienda XHTML, bajo el título de Modulación XHTML. En paralelo con esto, el W3C también trabajó en un nuevo lenguaje que no era compatible con los lenguajes HTML y XHTML anteriores, llamándolo XHTML2.

En el momento en que se detuvo la evolución de HTML en 1998, se especificaron y publicaron partes de la API para HTML desarrollada por proveedores de navegadores



con el nombre DOM Nivel 1 (en 1998) y DOM Nivel 2 Core y DOM Nivel 2 HTML (a partir de 2000 y culminando en 2003). Estos esfuerzos se agotaron, con algunas especificaciones DOM Nivel 3 publicadas en 2004, pero el grupo de trabajo se cerró antes de que se completaran todos los borradores de Nivel 3.

En 2003, la publicación de XForms, una tecnología que se posicionó como la próxima generación de formularios web, despertó un renovado interés en la evolución del propio HTML, en lugar de encontrar reemplazos para él. Este interés surgió de la constatación de que el despliegue de XML como tecnología web se limitaba a tecnologías completamente nuevas (como RSS y Atom posteriores), en lugar de reemplazar las tecnologías implementadas existentes (como HTML).

Una prueba de concepto para demostrar que era posible extender los formularios HTML4 para proporcionar muchas de las características que introdujo XForms 1.0, sin requerir que los navegadores implementaran motores de renderizado que eran incompatibles con las páginas web HTML existentes, fue el primer resultado de este renovado interés. En esta etapa inicial, aunque el borrador ya estaba disponible públicamente y ya se solicitaba información de todas las fuentes, la especificación solo estaba bajo los derechos de autor de Opera Software.

La idea de que la evolución de HTML debería reabrirse se probó en un taller del W3C en 2004, donde se presentaron algunos de los principios que subyacen al trabajo HTML5 (descrito a continuación), así como el borrador preliminar de la propuesta que cubre solo las características relacionadas con los formularios. el W3C conjuntamente por Mozilla y Opera. La propuesta fue rechazada debido a que entraba en conflicto con la dirección previamente elegida para la evolución de la Web; El personal y la membresía del W3C votaron para continuar desarrollando reemplazos basados en XML.

Poco después, Apple, Mozilla y Opera anunciaron conjuntamente su intención de continuar trabajando en el esfuerzo bajo el paraguas de un nuevo lugar llamado WHATWG. Se creó una lista de correo pública y el borrador se trasladó al sitio WHATWG. Posteriormente, los derechos de autor se modificaron para ser propiedad conjunta de los tres proveedores y permitir la reutilización de la especificación.

El WHATWG se basó en varios principios básicos, en particular que las tecnologías deben ser compatibles con versiones anteriores, que las especificaciones y las implementaciones deben coincidir incluso si esto significa cambiar la especificación en lugar de las implementaciones, y que las especificaciones deben ser lo suficientemente detalladas para que las implementaciones puedan lograr interoperabilidad completa sin ingeniería inversa entre sí.

El último requisito en particular requería que el alcance de la especificación HTML5 incluyera lo que previamente se había especificado en tres documentos separados:



HTML4, XHTML1 y DOM2 HTML. También significaba incluir significativamente más detalles de lo que previamente se había considerado la norma.

En 2006, el W3C mostró interés en participar en el desarrollo de HTML5 después de todo, y en 2007 formó un grupo de trabajo constituido para trabajar con el WHATWG en el desarrollo de la especificación HTML5. Apple, Mozilla y Opera permitieron que el W3C publicara la especificación bajo los derechos de autor del W3C, manteniendo una versión con la licencia menos restrictiva en el sitio WHATWG.

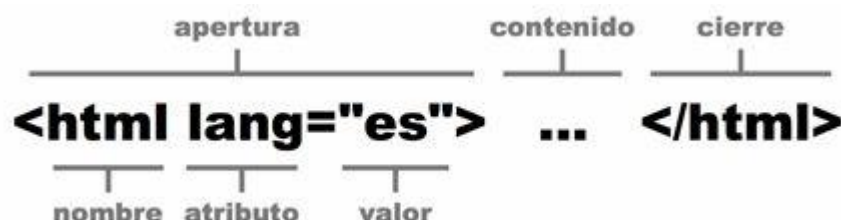
Durante varios años, ambos grupos trabajaron juntos. Sin embargo, en 2011, los grupos llegaron a la conclusión de que tenían objetivos diferentes: el W3C quería publicar una versión "terminada" de "HTML5", mientras que el WHATWG quería seguir trabajando en un estándar de vida para HTML, manteniendo continuamente la especificación en lugar de congelarlo en un estado con problemas conocidos y agregar nuevas características según sea necesario para evolucionar la plataforma.

En 2019, el WHATWG y el W3C firmaron un acuerdo para colaborar en una única versión de HTML en el futuro (<https://www.w3.org/2019/04/WHATWG-W3C-MOU.html>)

1.1.1.3. Estructura básica de un documento HTML.

Los documentos HTML consisten en un árbol de elementos y texto. Cada elemento se denota en la fuente mediante una etiqueta de inicio, como "<body>", y una etiqueta de finalización, como "</body>". (Ciertas etiquetas de inicio y finales pueden omitirse en ciertos casos y están implicadas por otras etiquetas).

La anatomía de uno de estos elementos, por ejemplo el caso de <html>, es típicamente de la siguiente forma^[7]:



De esta forma un documento HTML típico, está estructurado de la siguiente forma, como se muestra a continuación:



```
1  <!DOCTYPE html>
2  <html lang="es">
3  |   <head>
4  |   |   <title>Página de Ejemplo</title>
5  |   |   </head>
6  |   |   <body>
7  |   |   |   <h1>Página de Ejemplo</h1>
8  |   |   |   <p>Esta es un ejemplo <a href="demo.html">Simple</a>.</p>
9  |   |   |   <!-- Esto es un comentario -->
10 |   |   </body>
11 |   </html>
```

Las etiquetas deben estar anidadas de modo que todos los elementos estén complementados entre sí, sin superponerse:

```
<h2>Esto es <em>muy <strong>INCORRECTO</em> !</strong></h2>
```

```
<h2>Esto <em>es <strong>CORRECTO</strong>.</em></h2>
```

Esta especificación define un conjunto de elementos que se pueden usar en HTML, junto con reglas sobre las formas en que los elementos se pueden anidar.

Los elementos pueden tener atributos, que controlan cómo funcionan los elementos. En el ejemplo, hay un hipervínculo, formado utilizando el elemento **a** y su atributo **href**:

```
<a href="demo.html">Simple</a>
```

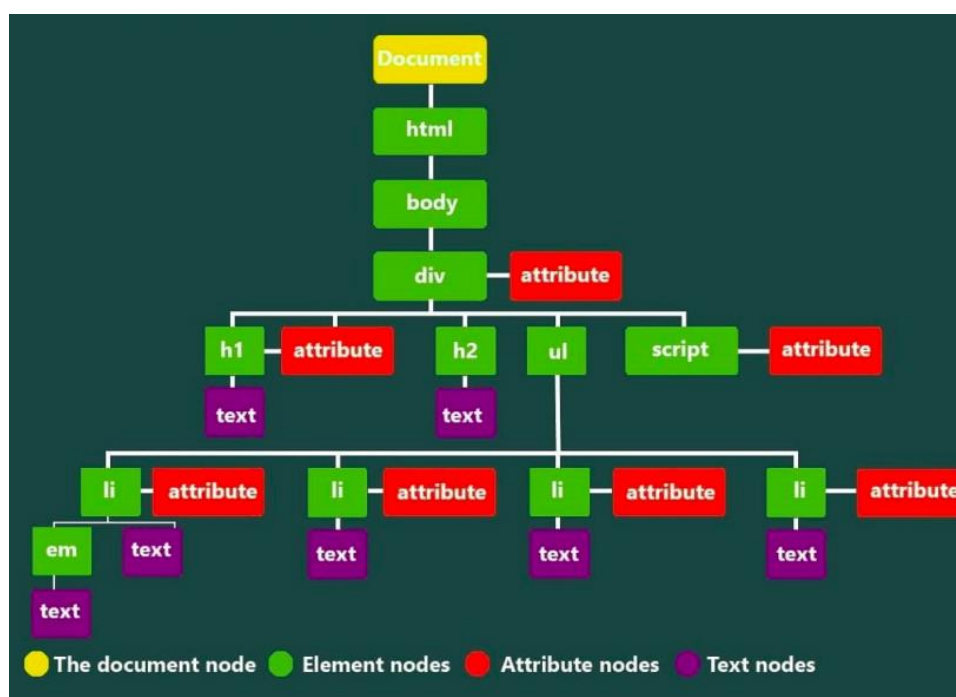
Los atributos se colocan dentro de la etiqueta de inicio y consisten en un nombre y un valor, separados por un carácter "=". El valor del atributo puede permanecer sin comillas si no contiene espacios en blanco ASCII o cualquiera de '"' = < ó >. De lo contrario, debe citarse con comillas simples o dobles. El valor, junto con el carácter "=", se puede omitir por completo si el valor es la cadena vacía.

```
<!-- atributos vacíos -->
<input name=address disabled>
<input name=address disabled="">

<!-- atributos con valor -->
<input name=address maxlength=200>
<input name=address maxlength='200'>
<input name=address maxlength="200">
```



Los agentes de usuario HTML (por ejemplo, los navegadores web) luego analizan este marcado, convirtiéndolo en un árbol DOM (Modelo de Objetos del Documento). Un árbol DOM es una representación en memoria de un documento y es una interfaz de programación para documentos HTML y XML. El DOM es una representación orientada a objetos de la página web, cuyos objetos y métodos se puede invocar y utilizar con un lenguaje de programación, tal como JavaScript e interactuar dinámicamente con el documento HTML/XML^[8].

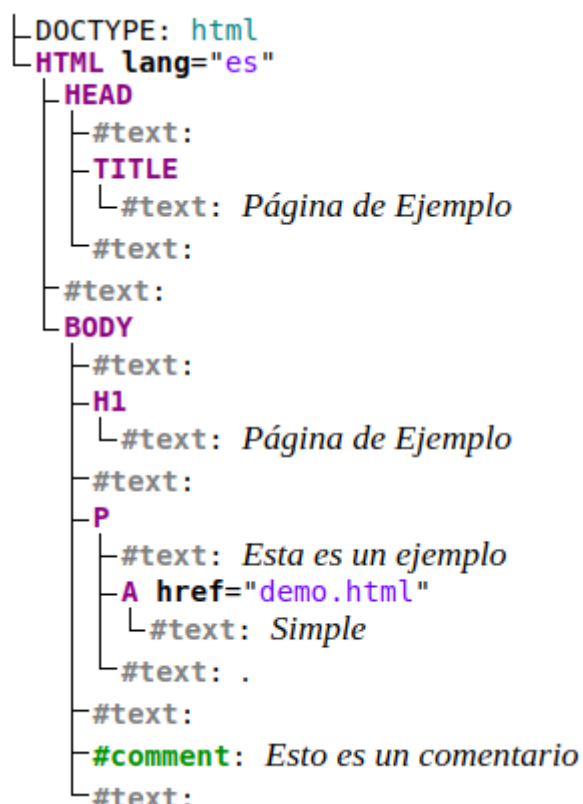


Árbol DOM

Fuente: <https://www.stefanocode.com/wp-content/uploads/2019/10/DOM-tree-example-1-1024x1024.jpg>

Los árboles DOM contienen varios tipos de nodos, en particular un nodo DocumentType, nodos Element, nodos Text, nodos Comment y, en algunos casos, nodos ProcessingInstruction.

El fragmento de marcado en la parte superior de esta sección se convertiría en el siguiente árbol DOM:



Visor online HTML a Árbol DOM, Fuente: <https://software.hixie.ch/utilities/js/live-dom-viewer/>

El elemento de documento de este árbol es el elemento **html**, que es el elemento que siempre se encuentra en esa posición en los documentos HTML. Contiene dos elementos, **head** y **body**, así como un nodo de **text** entre ellos.

Hay muchos más nodos de texto en el árbol DOM de los que uno esperaría inicialmente, porque el código fuente contiene varios espacios y saltos de línea que terminan como nodos de texto en el DOM. Sin embargo, por razones históricas, no todos los espacios y saltos de línea en el marcado original aparecen en el DOM. En particular, todo el espacio en blanco antes de **head** es descartado por el DOM, y todo espacio en blanco después de la etiqueta final de **body** se incluye al final de su contenido.

El elemento **head** contiene un elemento **title**, que contiene un nodo Text con el texto "Página de muestra". Del mismo modo, el elemento **body** contiene un elemento **h1**, un elemento **p** y un Comment.

Un nodo es el nombre genérico para cualquier tipo de objeto en la jerarquía DOM. Un nodo podría ser uno de los elementos nativos del DOM, como son **document** o



document.body. Podría ser una etiqueta especificada en HTML como **input** ó **p**; o podría ser un nodo de texto creado por el sistema para contener un bloque de texto dentro de otro elemento. Entonces, en pocas palabras, un nodo es cualquier objeto DOM.

El DOM consiste en una jerarquía de nodos donde cada nodo puede tener un padre, una lista de nodos hijos y un `nextSibling` y `previousSibling`. Esa estructura forma una jerarquía en forma de árbol. El nodo **document** tendría su lista de nodos hijos (el nodo **head** y el nodo **body**). El nodo **body**, a su vez, tendría su lista de nodos hijos y así sucesivamente.

Un elemento es un tipo específico de nodo, uno que puede especificarse directamente con una etiqueta HTML y puede tener propiedades como una `id` o una `clase`, puede tener hijos, etc. Hay otros tipos de nodos, como nodos de comentarios, nodos de texto, etc. con diferentes características. Cada nodo tiene una propiedad `.nodeType` que informa de qué tipo de nodo es^[9].

Este árbol DOM puede manipularse desde programas, o scripts, en la página. Las secuencias de comandos (generalmente en JavaScript, aunque como se muestra más adelante también es posible desde otros lenguajes) son pequeños programas que se pueden intercalar en el mismo documento HTML. Por ejemplo, este es un formulario con un script que establece el valor del elemento de salida del formulario para decir "Hello World"^[10]:

```
28 <form name="main">
29   Result: <output name="result"></output>
30   <script>
31     document.forms.main.elements.result.value = 'Hello World';
32   </script>
33 </form>
```

Cada elemento en el árbol DOM está representado por un objeto, y estos objetos tienen API para que puedan ser manipulados. Por ejemplo, un enlace puede cambiar su atributo `"href"` de varias maneras:

```
37 var a = document.links[0]; // obtiene el primer link de la página
38 a.href = 'sample.html'; // cambia la URL de destino del link
39 a.protocol = 'https'; // cambia el esquema de la URL
40 a.setAttribute('href', 'https://example.com/'); // cambia href
```



DOM Y LENGUAJES DE PROGRAMACIÓN

El breve ejemplo anterior está escrito en JavaScript, pero utiliza el DOM para acceder al documento y sus elementos. El DOM no es un lenguaje de programación, pero sin él, el lenguaje JavaScript no tendría ningún modelo o noción de páginas web, documentos HTML, documentos XML y sus componentes. Cada elemento de un documento (El documento en su conjunto, el encabezado, las tablas dentro del documento, los encabezados de las tablas, el texto dentro de las celdas de la tabla) forma parte del DOM para ese documento, por lo que se puede acceder a todos y manipularlos utilizándolo junto con un lenguaje de programación como JavaScript.

Al principio, JavaScript y el DOM estaban estrechamente entrelazados, pero finalmente evolucionaron como entidades separadas. El contenido de la página se almacena en el DOM y se puede acceder y manipular a través de JavaScript, pudiendo entenderlo como:

API = DOM + JavaScript

El DOM fue diseñado para ser independiente de un lenguaje de programación en particular, haciendo que la representación estructural del documento esté disponible desde una API única y consistente. Aunque en general nos centramos exclusivamente en JavaScript, las implementaciones del DOM se pueden construir para cualquier lenguaje, como demuestra este ejemplo de Python^[10]:

```
1 from xml.dom import minidom
2 doc = minidom.parse("staff.xml")
3 staffs = doc.getElementsByTagName("staff")
4 staffs[0].setAttribute("id", "1234")
```

CSS

Los documentos HTML representan una descripción de contenido interactivo, pero desde un punto de vista independiente de la presentación visual de éste. Los documentos HTML pueden mostrarse en una pantalla, a través de un sintetizador de voz o en una pantalla braille. Para influir exactamente en cómo se lleva a cabo dicha representación, los autores pueden usar un lenguaje de estilo como CSS.

En el siguiente ejemplo, la página se ha convertido en amarillo sobre azul con CSS.



```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <title>Sample styled page</title>
5      <style>
6        body { background: navy; color: yellow; }
7      </style>
8    </head>
9    <body>
10     <h1>Sample styled page</h1>
11     <p>This page is just a demo.</p>
12   </body>
13 </html>
```

1.1.1.4. El editor de código

Un editor de código es la herramienta esencial para escribir y comunicar instrucciones al intérprete o compilador de cierto lenguaje en el que se está desarrollando un software.

EDITORES Y PROCESADORES DE TEXTO

Hay muchas circunstancias diferentes bajo las cuales se debe trabajar con texto. Como tal, es importante reconocer los usos para los que están destinados los diferentes programas de manipulación de texto. Estos programas generalmente se dividen en 3 categorías: editores de texto e IDEs (Integrated Development Environments) y procesadores de texto. Exploraremos las diferencias y los usos de estos tipos de software a continuación. Cuando se trabaja con datos en archivos de texto, los editores de texto e IDEs suelen ser una mejor opción. Crean archivos no propietarios que se pueden transferir entre sistemas operativos sin la necesidad de un software intermedio. Además, tanto los archivos creados por los editores de texto como los propios editores de texto ocupan una gran cantidad de memoria en comparación con los procesadores de texto. Finalmente, con respecto a los datos, muchos programas permitirán que los archivos del editor de texto se utilicen como entrada, lo que ciertamente no es el caso con los archivos de procesador de texto, con su formato que generalmente los hace ilegibles para otros programas^{[11][12]}.

IDEs Y EDITORES DE TEXTO

El término **IDE** proviene de Integrated Development Environment. Está pensado como un conjunto de herramientas que funcionan en conjunto: editor de texto, compilador, build, make, integración, depuración, etc. Prácticamente todos los IDE están vinculados específicamente a un lenguaje o framework o conjunto de



lenguajes. Algunos ejemplos: Visual Studio para .NET y otros lenguajes de Microsoft, RubyMine para Ruby, IntelliJ para Java, XCode para tecnologías de Apple.

Un **Editor de Texto** es simplemente eso, una herramienta diseñada para editar texto plano. Técnicamente hablando, los únicos datos que contiene un archivo producido por un editor de texto son los valores que representan los caracteres individuales, que el programa muestra como los propios caracteres. El editor de texto es una herramienta básica de todos los sistemas operativos; Los usuarios de Windows probablemente estarán más familiarizados con el Bloc de Notas, aunque existen alternativas, como el editor de texto de código abierto, Notepad ++. Un editor de texto puede no ser ideal si tiene la intención de incluir algún tipo de formato en su texto, como la alineación; fuente o tamaño de letra; características de texto como negrita o cursiva; o la incorporación de elementos no textuales, como imágenes. Sin embargo, es posible formatear dicho documento utilizando varios lenguajes de marcado, como XML.

Por lo general, los editores están optimizados para lenguajes de programación, aunque muchos editores de texto de programadores se están diversificando y agregan características para texto que no es de programación como Markdown. La clave aquí es que los editores de texto están diseñados para funcionar con cualquier lenguaje o framework que escojamos y que sea soportado.

Si bien generalmente la curva de aprendizaje es más corta si se está trabajando en el ámbito de un IDE, a largo plazo se invierte más tiempo al querer trabajar con otros lenguajes o frameworks y sus correspondientes IDEs. Si usa un Editor de Texto, puede es posible mantener los mismos flujos de trabajo. Las herramientas que se hayan ido incorporando al editor se pueden transferir al siguiente lenguaje y framework. Su editor se vuelve más potente y más personalizado a la manera en que se desea trabajar, no solo durante años, sino potencialmente décadas. Existen desarrolladores que utilizan Vim o Emacs, editores que han estado disponibles por más de 25 años.

Como conclusión, frente a la necesidad de ser productivo de inmediato en una tecnología específica, tal vez un IDE sea apropiado. Si se desea una herramienta que pueda ser moldeada y personalizada exactamente según lo que se desee, para trabajar con múltiplease lenguajes y frameworks, entonces un editor es probablemente lo más apropiado.



PROCESADORES DE TEXTO

A diferencia de un editor de texto, un procesador de texto es cualquier programa a través del cual el texto (y, a menudo, otros tipos de medios) se puede formatear y preparar para imprimir, ya sea físico o electrónico. Estos le dan al usuario un amplio control sobre las cualidades visuales del documento. Los archivos de este tipo, aunque son preferidos por lectores humanos, generalmente no son adecuados para ser procesados por computadores, en contenidos como un fragmento de código o una lista de valores.

Para ilustrar la diferencia entre documentos creados por editores de texto y procesadores de texto en un nivel muy básico, podemos abrir los archivos en un editor de texto. Un archivo creado por un editor de texto, independientemente de si era el mismo editor de texto o no, simplemente mostrará el contenido de texto del documento. Sin embargo, al intentar abrir un documento creado por un procesador de textos, puede esperar ver algo como lo que se muestra a continuación, que simplemente es el contenido mezclado con muchas directivas propietarias del procesador de texto en particular para el formato:

```

1  PK    ! d<n™      [Content_Types].xml  ¢(
2  •ËNÄ0E÷HÜCä-J\X „š²à±$ŠX{ØX$¶āý{&MjqytÉ±çPā±u=>
3  {k>ì"ZīJqXEDN{cÝ~÷Ó«üDdHÉÖx¥~Š³ÉpPx:€W; ,EMN¥D]C«
4  °ðİT>¶Šxg2(ý¶f F£c@½#p"Š§!&ā "ÔsCÜāÿîI"4(²ó~açU
5  Bcµ"&•/İ|qÉ-W.Ö`m0†k™İĖ°nM´²[éZµ€!_]40xýÜòŠieÖp
6  ú³²†úN-D´´{P6Ā00*ëVü9æ
7  àÿSô°[Ú?X³/«
8  4v°-æÝ| <PâCmÚ
9  ^,İÜ~|¾,yæé,TN"¾ĀāÿÎ(>^'A*İEyÚÀÜ0IpFG +ā$B
10 Ê@<ÜâPýðJôĀİÿŠ#j³ø,A:   Aæ» B¿wb!ó%'Ômô9Çă/¶½
11 ê®:çè ÉĀŌëçnpā7àİ}†î•1`ÖxĒĀ«6y ỹỹ PK    ! '·ó  N
12 |   _rels/.rels ¢(

```

1.1.1.5. Editores recomendados.

Los documentos HTML, así como los archivos CSS y JavaScript, son archivos de texto, por lo que podemos usar cualquier editor incluido en nuestro ordenador para crearlos, como el bloc de notas de Windows o la aplicación editor de texto de los ordenadores de Apple, pero también existen editores de texto especialmente diseñados para programadores y desarrolladores web que pueden simplificar nuestro trabajo. Estos editores resaltan texto con diferentes colores para ayudarnos a identificar cada parte del código, o listan los archivos de un proyecto en un panel lateral para ayudarnos a trabajar con múltiples archivos al mismo tiempo. La siguiente es una lista de los editores y de IDE (Integrated Development Environments) más populares disponibles para ordenadores personales y ordenadores de Apple.



- **Visual Studio Code** (code.visualstudio.com) es un editor gratuito, producido por Microsoft pero de código abierto (**Recomendado**).
- **Atom** (www.atom.io) es un editor gratuito, simple de usar y altamente personalizable (**Recomendado**).
- **Sublime** (www.sublimetext.com) es un editor de pago con una versión gratuita de evaluación.
- **Notepad ++** (notepad-plus-plus.org) es un editor de código fuente gratuito y reemplazo de Bloc de notas para MS windows que admite múltiples lenguajes.

Trabajar con un editor es simple: tenemos que crear un directorio en nuestro disco duro donde vamos a almacenar los archivos del sitio web, abrir el editor, y crear dentro de este directorio todos los archivos y directorios adicionales que necesitamos para nuestro proyecto.

Funciones básicas que se esperan en un editor de texto para programación son, en distinto grado de facilidades, las siguientes:

Característica	Descripción
Resaltado de sintaxis	Establece las diferentes partes del código en diferentes colores para indicar lo que representan. Una característica muy común en los editores de código, el resaltado de sintaxis se puede utilizar para mejorar la legibilidad y el contexto del código, y también ayudarlo a encontrar errores. Los editores tienen la facilidad de detectar el lenguaje en que está escrito determinado código y resaltarán la sintaxis que corresponda según éste.
Ejecución de Código y Previsualización	Le permite ver cómo se verá una página web en la misma ventana del editor de código en la que está trabajando (en lugar de ir y venir entre el editor de código y el navegador) o ejecutar el código de un lenguaje de programación. Algunos editores de código tienen la opción de alternar entre "vista de código" y "vista previa", mientras que otros le permiten ver su código y página al mismo tiempo.
Zoom de Texto	Permite aumentar o disminuir el tamaño del código.
Plugins	Un complemento es un componente que aumenta la funcionalidad de un programa. Con algunos editores de código, puede instalar complementos que mejorarán su experiencia de codificación. Por ejemplo, un editor de texto puede tener un complemento FTP que se puede utilizar para integrar la funcionalidad FTP directamente en él.
Capacidades de búsqueda avanzada	Muchos editores de código tienen capacidades de búsqueda avanzadas que le permiten hacer cosas como



	buscar por mayúscula (mayúscula o no mayúscula), buscar múltiples archivos a la vez, realizar búsquedas de expresiones regulares y más.
Multivisualización	Algunos editores de código tienen una opción de vista múltiple que le permite ver el código de dos documentos uno al lado del otro.
Autocomplete de código	Al comenzar a escribir algún código, cerca de éste aparece una pequeña lista desplegable que da la opción de cómo terminar lo que acaba de comenzar a escribir.
Tabs	En caso de trabajar en más de un archivo a la vez, utilizando pestañas se pueden ver tantos archivos en una ventana como desee y cada archivo se abrirá en una pestaña separada.
Resaltado de bloques de código	Cuanto más complejo es el código, más difícil es realizar un seguimiento. Con el resaltado de bloque de código, el editor de código informa dónde comienza y dónde termina un bloque de código.
Terminal	Es común que un editor de texto tenga la posibilidad de mostrar una sub-ventana donde el usuario pueda interactuar con el prompt del sistema operativo y ejecutar comandos en éste.

VISUAL STUDIO CODE



Url: <https://code.visualstudio.com/>

Desarrollador: Microsoft

Plataformas: OSX, Windows, Linux

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto, aunque la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft.

Visual Studio Code se basa en Electron, un framework que se utiliza para implementar Chromium y Node.js como aplicaciones para escritorio, que se ejecuta en el motor de diseño Blink. Aunque utiliza el framework Electron, el software no usa Atom y en su lugar emplea el mismo componente editor (Monaco) utilizado en Visual Studio Team Services (anteriormente llamado Visual Studio Online).



Visual Studio Code es un editor de código fuente. Es compatible con varios lenguajes de programación y un conjunto de características que pueden o no estar disponibles para un idioma dado, como se muestra en la siguiente tabla. Muchas de las características de Visual Studio Code no están expuestas a través de los menús o la interfaz de usuario. Más bien, se accede a través de la paleta de comandos o a través de archivos .json (por ejemplo, preferencias del usuario). La paleta de comandos es una interfaz de línea de comandos. Sin embargo, desaparece si el usuario hace clic fuera de él o presiona una combinación de teclas en el teclado para interactuar con algo que está fuera de él. Esto también se aplica a los comandos que requieren mucho tiempo. Cuando esto sucede, el comando en progreso se cancela.

En el rol de editor de código fuente, Visual Studio Code permite cambiar la página de códigos en la que se guarda el documento activo, el carácter que identifica el salto de línea (una opción entre LF y CRLF) y el lenguaje de programación del documento activo.

```
landing.html - mgclients - Visual Studio Code

EXPLORER
  app.py
  clients.py
  landing.html X
  app.py
  clients.py a... M
  landing.html app...

MGCLIENTS
  __pycache__
  .vscode
  app_mg
  forms
  models
  static
  templates
  clients
  error
  keep
  macro
  operations
  resumen
  users
  data_success.html
  example.html
  landing.html
  OUTLINE
  TIMELINE

app_mg > templates > landing.html > html > head
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8"/>
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <meta name="description" content="Exchange rates, intercambio de divisas">
8   <meta name="author" content="Alfanetics">
9   <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/boo
10  <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/gek
11  <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/fon
12  <!-- styles dev diego -->
13 </head>
14
15 <body background="{{ url_for('static', filename='images/back.jpeg') }}">
16
17 <div class="container">
```

Visual Studio Code se puede extender a través de complementos, disponible a través de un repositorio central. Esto incluye adiciones al editor y soporte de idiomas. Una característica notable es la capacidad de crear extensiones que analizan código, como linters y herramientas para análisis estático, utilizando el Protocolo de Servidor de Idioma.

ATOM



Url: <https://atom.io/>

Desarrollador: GitHub Inc.

Plataformas: OSX, Windows, Linux

Llamado por su desarrollador como un “Editor de texto hackable para el siglo XI”.

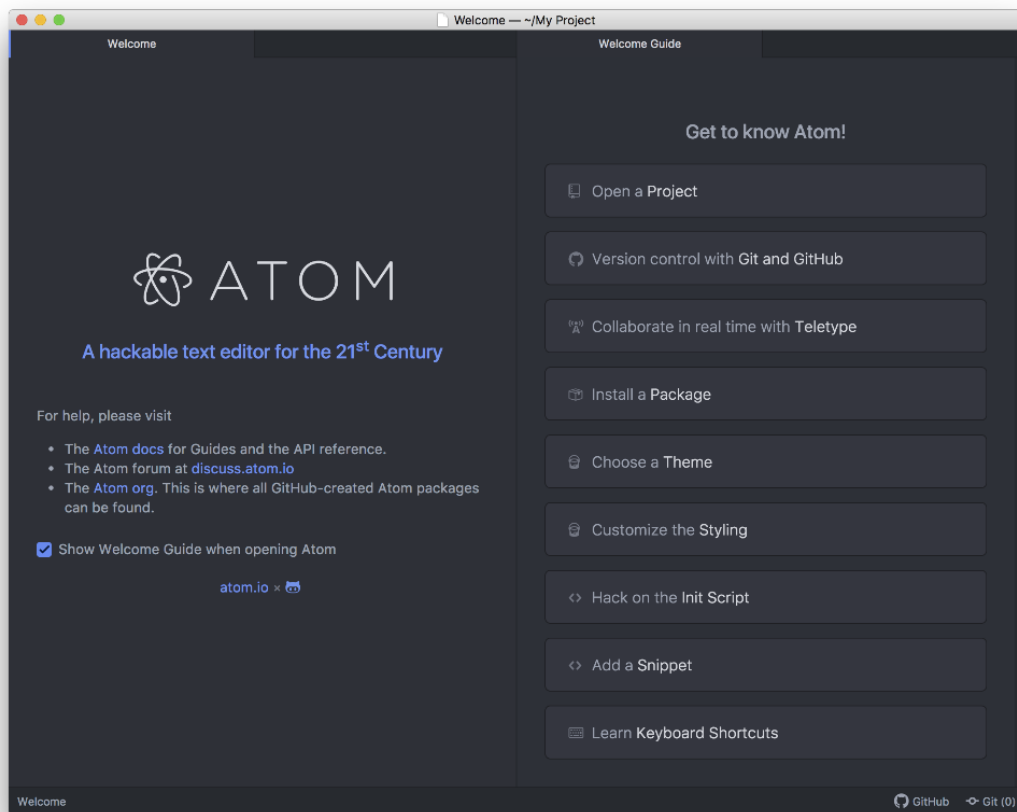


Atom es un editor de código fuente de código abierto para macOS, Linux, y Windows, con soporte para múltiples plug-in escritos en Node.js y control de versiones Git integrado, desarrollado por GitHub. Atom es una aplicación de escritorio construida utilizando tecnologías web.

La mayor parte de los paquetes tienen licencias de software libre y está desarrollados y mantenidos por la comunidad de usuarios. Atom está basado en Electron (Anteriormente conocido como Atom Shell), un framework que permite crear aplicaciones de escritorio multiplataforma usando Chromium y Node.js. Está escrito en CoffeeScript y Less. También puede ser utilizado como un entorno de desarrollo integrado (IDE). Atom liberó su beta en la versión 1.0, el 25 de junio de 2015.

Atom puede añadir soporte para otros lenguajes de programación mediante el sistema de paquetes, así también mejorar el soporte para los lenguajes existentes mediante mejoras como intérpretes, debuggers o pipelines que conecten software de terceros a Atom. El administrador de paquetes se instala de manera predeterminada y, para mejorar aún más, todos los paquetes están alojados en Github.

Al momento de ser lanzado Atom aproximadamente 6,500 paquetes y temas disponibles. Los paquetes son tan fundamentales para la filosofía de Atom que las características principales, como la vista de árbol y la vista de configuración, son simplemente paquetes preinstalados.



Git está desarrollado por Github y su integración con esta plataforma online para almacenamiento y administración de repositorios en línea es muy buena.

Atom posee una gran documentación para introducirse y aprender todas sus facilidades: <https://flight-manual.atom.io/>

SUBLIME TEXT



Url: <https://www.sublimetext.com/>

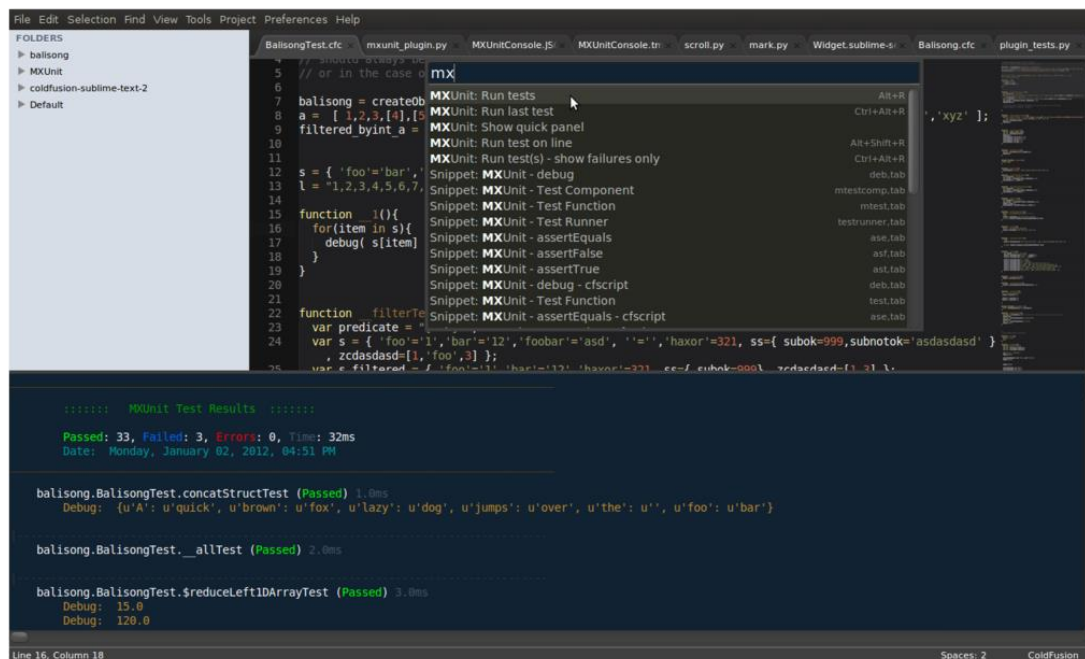
Desarrollador: Jon Skinner

Plataformas: OSX, Windows, Linux

Sublime Text es un editor de texto y editor de código fuente está escrito en C++ y Python para los plugins. Desarrollado originalmente como una extensión de Vim, con el tiempo fue creando una identidad propia, por esto aún conserva un modo de edición tipo vi llamado Vintage mode.



Se puede descargar y evaluar de forma gratuita. Sin embargo no es software libre o de código abierto y se debe obtener una licencia para su uso continuado, aunque la versión de evaluación es plenamente funcional y no tiene fecha de caducidad.



NOTEPAD ++



Url: <https://notepad-plus-plus.org/>

Desarrollador: Don Ho

Plataformas: Microsoft Windows

Editor de código fuente con soporte para diversos lenguajes de programación, gratuito y de código libre.

Notepad++ es un programa para editar código fuente de cualquier lenguaje de programación. Como tiene soporte para una gran cantidad de lenguajes, interesará no sólo a los desarrolladores de webs, sino en general a toda la comunidad de programadores.

Es de estos editores que ofrecen ayudas muy útiles para "tirar líneas de código", como resaltado de colores, posibilidad de editar varios documentos a la vez, menús contextuales, auto-completar código, etc. Todo un regalo para los programadores, ya que además es gratuito.



Es un proyecto creado a partir de otro editor para programadores llamado Scintilla. Por decirlo de alguna manera, Notepad++ es una distribución de Scintilla, pero con algunas contribuciones adicionales.

Se distribuye sólo para sistemas Windows y está programado en C++, utilizando directamente el API de win32, lo que hace que sea rápido y con un archivo de descarga pequeño. Se puede descargar y conocer más sobre el programa en: <https://notepad-plus-plus.org>


```
1 // MediaWiki
2
3 MediaWiki is a free and open-source wiki software package written in PHP. It
4 serves as the platform for Wikipedia and the other Wikimedia projects, used
5 by hundreds of millions of people each month. MediaWiki is localized in over
6 100 languages and its reliability and robust feature set have earned it a large
7 and vibrant community of third-party users and developers.
8
9 MediaWiki is:
10
11 * Feature-rich and extensible, both on-wiki and with hundreds of extensions;
12 * Scalable and suitable for both small and large sites;
13 * Simple to install, working on most hardware/software combinations; and
14 * Available in your language.
15
16 For system requirements, installation, and upgrade details, see the files
17 RELEASE-NOTES, INSTALL, and UPGRADE.
18
19 ** Ready to get started?
20 ** https://www.mediawiki.org/wiki/Download
21 ** Looking for the technical manual?
22 ** https://www.mediawiki.org/wiki/Manual:Contents
23 ** Seeking help from a person?
24 ** https://www.mediawiki.org/wiki/Communication
25 ** Looking to file a bug report or a feature request?
26 ** https://bugzilla.mediawiki.org/
27 ** Interested in helping out?
28 ** https://www.mediawiki.org/wiki/How_to_contribute
29
30 MediaWiki is the result of global collaboration and cooperation. The CREDITS
31 file lists technical contributors to the project. The COPYING file explains
32 MediaWiki's copyright and license (GNU General Public License, version 2 or
33 later). Many thanks to the Wikimedia community for testing and suggestions.
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

1.1.1.6. Etiquetas HTML5

Existe una variada documentación, tutoriales, resúmenes y cheatsheets en internet que se han acumulado con el correr de los años. Esto muchas veces confunden al desarrollador en relación a cuáles son las etiquetas actualmente válidas, y cuáles están obsoletas. De la misma forma, cuáles corresponden a nuevas etiquetas introducidas en HTML5 y cuáles son herencias de versiones anteriores. Esto puede parecer un simple dato, sin embargo es bueno tener presente que en este campo cambiante de tecnologías de punta debemos estar permanente preocupados de cuál es el grado de compatibilidad de los navegadores web respecto de nuevos estándares. Navegadores antiguos eventualmente presenten problemas con nuevos estándares y eso debe considerarse al momento de desarrollar y considerar el perfil del usuario típico de nuestro sistema web.

Por lo anterior, y con el objetivo de proveer una resumen claro y útil de referencia para el desarrollador, entregamos a continuación un listado de todos los elementos del estandar HTML5, descritos por su etiqueta de apertura y agrupados por su



función. Se indica con un símbolo  las etiquetas que han sido introducidas en la versión 5 del estándar HTML^[13].

ELEMENTO RAIZ

Son etiquetas básicas de la jerarquía más alta para definir y estructurar el documento HTML.

Elemento	Descripción
<u><!doctype html></u>	Define que el documento esta bajo el estandar de HTML 5

Elemento	Descripción
<u><html></u>	Representa la raíz de un documento HTML o XHTML. Todos los demás elementos deben ser descendientes de este elemento.

METADATOS DEL DOCUMENTO

Los contenidos metadato se caracterizan por contener información de tipo metadato, los metadatos determinan la manera en la que se comportan, actúan, aparecen, los elementos del documento HTML.

Además estos contenidos metadatos generalmente están en el encabezado y también son capaces de establecer comunicación con otros documentos relacionados o externos.

Elemento	Descripción
<u><head></u>	Representa una colección de metadatos acerca del documento, incluyendo enlaces a, o definiciones de, scripts y hojas de estilo.
<u><title></u>	Define el título del documento, el cual se muestra en la barra de título del navegador o en las pestañas de página. Solamente puede contener texto y cualquier otra etiqueta contenida no será interpretada.
<u><base></u>	Define la URL base para las URLs relativas en la página.
<u><link></u>	Usada para enlazar JavaScript y CSS externos con el documento HTML actual.
<u><meta></u>	Define los metadatos que no pueden ser definidos usando otro elemento HTML.
<u><style></u>	Etiqueta de estilo usada para escribir CSS en línea.



SCRIPTING

Se refiere a etiquetas que permiten crear elementos HTML de tipo script, que pueden agregar funcionalidades dinámicas al documento a través, por ejemplo, de manipulación de la API del DOM.

Elemento	Descripción
<u><script></u>	Define ya sea un script interno o un enlace hacia un script externo. El lenguaje de programación es JavaScript
<u><noscript></u>	Define un contenido alternativo a mostrar cuando el navegador no soporta scripting.

SECCIONES



Los elementos de seccionamiento del contenido permiten organizar los contenidos del documento en partes lógicas. Usa los elementos de seccionado para crear una descripción amplia de los contenidos del documento, incluyendo el encabezado y pie de página y elementos para identificar secciones.

Elemento	Descripción
<u><body></u>	Representa el contenido principal de un documento HTML. Solo hay un elemento <body> en un documento.
<u><section></u> 	Define una sección en un documento.
<u><nav></u> 	Define una sección que solamente contiene enlaces de navegación
<u><article></u> 	Define contenido autónomo que podría existir independientemente del resto del contenido.
<u><aside></u> 	Define algunos contenidos vagamente relacionados con el resto del contenido de la página. Si es removido, el contenido restante seguirá teniendo sentido
<u><h1>,<h2>,<h3>,<h4>,<h5>,<h6></u> <u>></u>	Los elemento de cabecera implementan seis niveles de cabeceras de documentos; <h1> es la de mayor y <h6> es la de menor importancia. Un elemento de cabecera describe brevemente el tema de la sección que introduce.
<u><header></u> 	Define la cabecera de una página o sección. Usualmente contiene un logotipo, el título del sitio Web y una tabla de navegación de contenidos.
<u><footer></u> 	Define el pie de una página o sección. Usualmente contiene un mensaje de derechos de autoría, algunos enlaces a información legal o direcciones para dar información de retroalimentación.
<u><address></u>	Define una sección que contiene información de contacto.
<u><main></u> 	Define el contenido principal o importante en el documento. Solamente existe un elemento <main> en el documento.



AGRUPACIÓN DE CONTENIDO

Son etiquetas que permiten agrupar el contenido en un documento, tales como párrafos, listas, contenedores y otras.








Elemento	Descripción
<u><p></u>	Define una parte que debe mostrarse como un párrafo.
<u><hr></u>	Representa un quiebre temático entre párrafos de una sección o artículo o cualquier contenido.
<u><pre></u>	Indica que su contenido está preformateado y que este formato debe ser preservado.
<u><blockquote></u>	Representa un contenido citado desde otra fuente.
<u></u>	Define una lista ordenada de artículos.
<u></u>	Define una lista de artículos sin orden.
<u></u>	Define un artículo de una lista enumerada.
<u><dl></u>	Define una lista de definiciones, es decir, una lista de términos y sus definiciones asociadas.
<u><dt></u>	Representa un término definido por el siguiente <code><dd></code> .
<u><dd></u>	Representa la definición de los términos listados antes que él.
<u><figure></u> 	Representa una figura ilustrada como parte del documento.
<u><figcaption></u> 	Representa la leyenda de una figura.
<u><div></u>	Representa un contenedor genérico sin ningún significado especial.

SEMÁNTICA A NIVEL DE TEXTO


Estas etiquetas definen el significado, estructura, o el estilo de una palabra, una línea o cualquier pieza arbitraria de texto.

Elemento	Descripción
<u><a></u>	Representa un <i>hiper enlace</i> , enlazando a otro recurso.
<u></u>	Representa un texto <i>enfático</i> , como un acento de intensidad.
<u></u>	Representa un texto especialmente <i>importante</i> .
<u><small></u>	Representa un <i>comentario aparte</i> , es decir, textos como un descargo de responsabilidad o una nota de derechos de autoría, que no son esenciales para la comprensión del documento.
<u><s></u>	Representa contenido que <i>ya no es exacto o relevante</i> .
<u><cite></u>	Representa el <i>título de una obra</i> .
<u><q></u>	Representa una <i>cita textual</i> inline.
<u><dfn></u>	Representa un término cuya <i>definición</i> está contenida en su contenido ancestro más próximo.
<u><abbr></u>	Representa una <i>abreviación</i> o un <i>acrónimo</i> ; la



	expansión de la abreviatura puede ser representada por el atributo title.
<u><data></u> 	Asocia un <i>equivalente legible por máquina</i> a sus contenidos. (Este elemento está solamente en la versión de la WHATWG del estandar HTML, y no en la versión de la W3C de HTML5).
<u><time></u> 	Representa un valor de <i>fecha y hora</i> ; el equivalente legible por máquina puede ser representado en el atributo datetime.
<u><code></u>	Representa un <i>código de ordenador</i> .
<u><var></u>	Representa a una <i>variable</i> , es decir, una <i>expresión matemática o contexto de programación</i> , un <i>identificador que represente a una constante</i> , un <i>símbolo que identifica una cantidad física</i> , un <i>parámetro de una función o un marcador de posición en prosa</i> .
<u><samp></u>	Representa la <i>salida</i> de un programa o un ordenador.
<u><kbd></u>	Representa la <i>entrada de usuario</i> , por lo general desde un teclado, pero no necesariamente, este puede representar otras formas de entrada de usuario, como comandos de voz transcritos.
<u><sub></u> , <u><sup></u>	Representan un <i>subíndice</i> y un <i>superíndice</i> , respectivamente.
<u><i></u>	Representa un texto en una voz o estado de ánimo <i>alterno</i> , o por lo menos de diferente calidad, como una designación taxonómica, un término técnico, una frase idiomática, un pensamiento o el nombre de un barco.
<u></u>	Representa un texto hacia el cual se llama la atención para <i>propósitos utilitarios</i> . No confiere ninguna importancia adicional y no implica una voz alterna.
<u><u></u>	Representa una anotación no textual <i>sin-articular</i> , como etiquetar un texto como mal escrito o etiquetar un nombre propio en texto en chino.
<u><mark></u> 	Representa texto resaltado con propósitos de <i>referencia</i> , es decir por su relevancia en otro contexto.
<u><ruby></u> 	Representa contenidos a ser marcados con <i>anotaciones ruby</i> , recorridos cortos de texto presentados junto al texto. Estos son utilizados con regularidad en conjunto a lenguajes de Asia del Este, donde las anotaciones actúan como una guía para la pronunciación, como el <i>furigana</i> japonés.
<u><rt></u> 	Representa el <i>texto de una anotación ruby</i> .
<u><rp></u> 	Representa los <i>paréntesis</i> alrededor de una anotación ruby, usada para mostrar la anotación de manera alterna por los navegadores que no soporten despliegue estandar para las anotaciones.
<u><bdi></u> 	Representa un texto que debe ser <i>aislado</i> de sus alrededores para el formateado bidireccional del texto.








	Permite incrustar un fragmento de texto con una direccionalidad diferente o desconocida.
<u><bdo></u>	Representa la <i>direccionalidad</i> de sus descendientes con el fin de anular de forma explícita al algoritmo bidireccional Unicode.
<u></u>	Representa texto sin un significado específico. Este debe ser usado cuando <i>ningún otro</i> elemento semántico le confiere un significado adecuado, en cuyo caso, provendrá de atributos globales como class, lang, o dir.
<u>
</u>	Representa un <i>salto de línea</i> .
<u><wbr></u> 	Representa una <i>oportunidad de salto de línea</i> , es decir, un punto sugerido de envoltura donde el texto de múltiples líneas puede ser dividido para mejorar su legibilidad.

EDICIONES




Elemento	Descripción
<u><ins></u>	Define una <i>adición</i> en el documento.
<u></u>	Define una <i>remoción</i> del documento.

CONTENIDO INCRUSTADO

Son etiquetas que permiten insertar en el documento HTML contenido Web, Videos, Audio y otros, provenientes de fuentes externas y/u otras páginas.

Elemento	Descripción
<u></u>	Representa una <i>imagen</i> .
<u><iframe></u>	Representa un <i>contexto anidado de navegación</i> , es decir, un documento HTML embebido.
<u><embed></u> 	Representa un <i>punto de integración</i> para una aplicación o contenido interactivo externo que por lo general no es HTML.
<u><object></u>	Representa un <i>recurso externo</i> , que será tratado como una imagen, un sub-documento HTML o un recurso externo a ser procesado por un plugin.
<u><param></u>	Define <i>parámetros</i> para el uso por los plugins invocados por los elementos <object>.
<u><video></u> 	Representa un <i>video</i> , y sus archivos de audio y canciones asociadas, con la interfaz necesaria para reproducirlos.
<u><audio></u> 	Representa un <i>sonido</i> o <i>stream de audio</i> .
<u><source></u> 	Permite a autores especificar recursos multimedia alternativos para los elementos multimedia como <video> o <audio>.
<u><track></u> 	Permite a autores especificar una <i>pista de texto</i> temporizado para elementos multimedia como



	<video> o <audio>.
<u><canvas></u> 	Representa un <i>área de mapa de bits</i> en el que se pueden utilizar scripts para renderizar gráficos como gráficas, gráficas de juegos o cualquier imagen visual al vuelo.
<u><map></u>	En conjunto con <area>, define un <i>mapa de imagen</i> .
<u><area></u>	En conjunto con <map>, define un <i>mapa de imagen</i> .
<u><svg></u> 	Define una <i>imagen vectorial</i> embebida.
<u><math></u> 	Define una <i>fórmula matemática</i> .

DATOS TABULARES

Corresponden a etiquetas que permiten ordenar contenidos en formato de tablas, con secciones, filas y columnas.






Elemento	Descripción
<u><table></u>	Representa <i>datos con más de una dimensión</i> .
<u><caption></u>	Representa el <i>título de una tabla</i> .
<u><colgroup></u>	Representa un <i>conjunto de una o más columnas</i> de una tabla.
<u><col></u>	Representa una <i>columna</i> de una tabla.
<u><tbody></u>	Representa el bloque de filas que describen los <i>datos concretos</i> de una tabla.
<u><thead></u>	Representa el bloque de filas que describen las <i>etiquetas de columna</i> de una tabla.
<u><tfoot></u>	Representa los bloques de filas que describen los <i>resúmenes de columna</i> de una tabla.
<u><tr></u>	Representa una <i>fila de celdas</i> en una tabla.
<u><td></u>	Representa una <i>celda de datos</i> en una tabla.
<u><th></u>	Representa una <i>celda encabezado</i> en una tabla.

FORMULARIOS

HTML provee un número de etiquetas que pueden usarse conjuntamente para crear formularios los cuales el usuario puede completar y enviar al sitio Web o a una aplicación.





Elemento	Descripción
<u><form></u>	Representa un <i>formulario</i> , consistiendo de controles que puede ser enviado a un servidor para procesamiento.
<u><fieldset></u>	Representa un <i>conjunto de controles</i> .
<u><legend></u>	Representa el <i>título</i> de un <fieldset>.
<u><label></u>	Representa el <i>título</i> de un control de formulario.
<u><input></u>	Representa un <i>campo de datos escrito</i> que permite al usuario editar los datos.
<u><button></u>	Representa un <i>botón</i> .
<u><select></u>	Representa un control que permite la <i>selección entre</i>



	<i>un conjunto de opciones.</i>
<code><datalist></code> 	Representa un <i>conjunto de opciones predefinidas</i> para otros controles.
<code><optgroup></code>	Representa un <i>conjunto de opciones</i> , agrupadas lógicamente.
<code><option></code>	Representa una <i>opción</i> en un elemento <code><select></code> , o una sugerencia de un elemento <code><datalist></code> .
<code><textarea></code>	Representa un <i>control de edición de texto multilínea</i> .
<code><keygen></code> 	Representa un control de <i>par generador de llaves</i> .
<code><output></code> 	Representa el <i>resultado de un cálculo</i> .
<code><progress></code> 	Representa el <i>progreso de finalización</i> de una tarea.
<code><meter></code> 	Representa la <i>medida</i> escalar (o el valor fraccionario) dentro de un rango conocido.

ELEMENTOS INTERACTIVOS

En esta categoría podemos encontrar a aquellos creados para la interacción con el usuario, tales como los enlaces, botones, campos de entrada de texto, audios, vídeos, etc.

Elemento	Descripción
<code><details></code> 	Representa un <i>widget</i> desde el que un usuario puede obtener información o controles adicionales.
<code><summary></code> 	Representa un <i>resumen, título o leyenda</i> para un elemento <code><details></code> dado.
<code><command></code> 	Representa un <i>comando</i> que un usuario puede invocar.
<code><menu></code> 	Representa una <i>lista de comandos</i> .

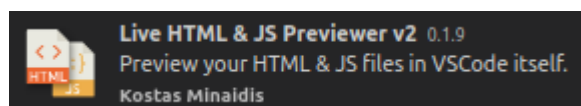
1.1.1.7. Creando un archivo HTML

A continuación abordaremos de una forma más práctica la creación de un archivo HTML para un objetivo simple, y aplicando algunas de las etiquetas expuestas en la sección anterior.

La idea de este punto es dar una visión de cómo se da partida a un proyecto Web, partiendo de su unidad básica que es su archivo HTML principal.

Utilizaremos en este caso el editor Visual Studio Code, que es uno de los que hemos mencionado como recomendado en la sección 1.1.1.5.

Abrimos el editor Visual Studio Code e instalamos la extensión Live HTML & JS Previewer v2:

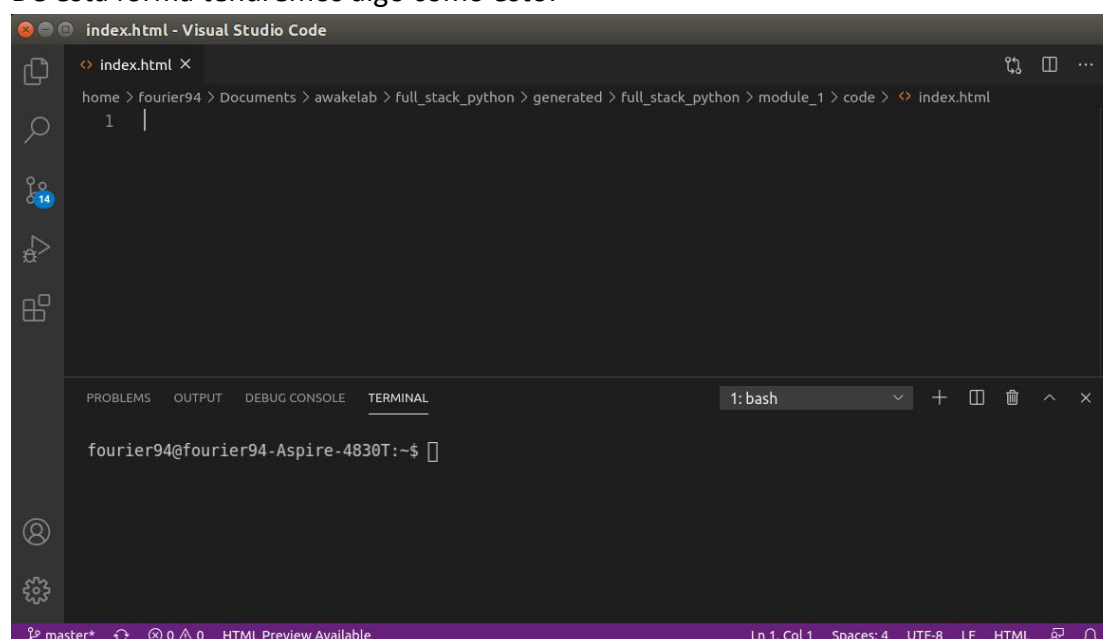


A través de esta extensión podremos ver instantáneamente, a medida que vayamos escribiendo nuestro código, el documento resultante HTML tal como se mostraría en un Navegador típico. Con esta extensión evitaremos tener que estar conmutando permanentemente a nuestro navegador y recargar la página cada vez que queramos ver los resultados de nuestra edición de código.

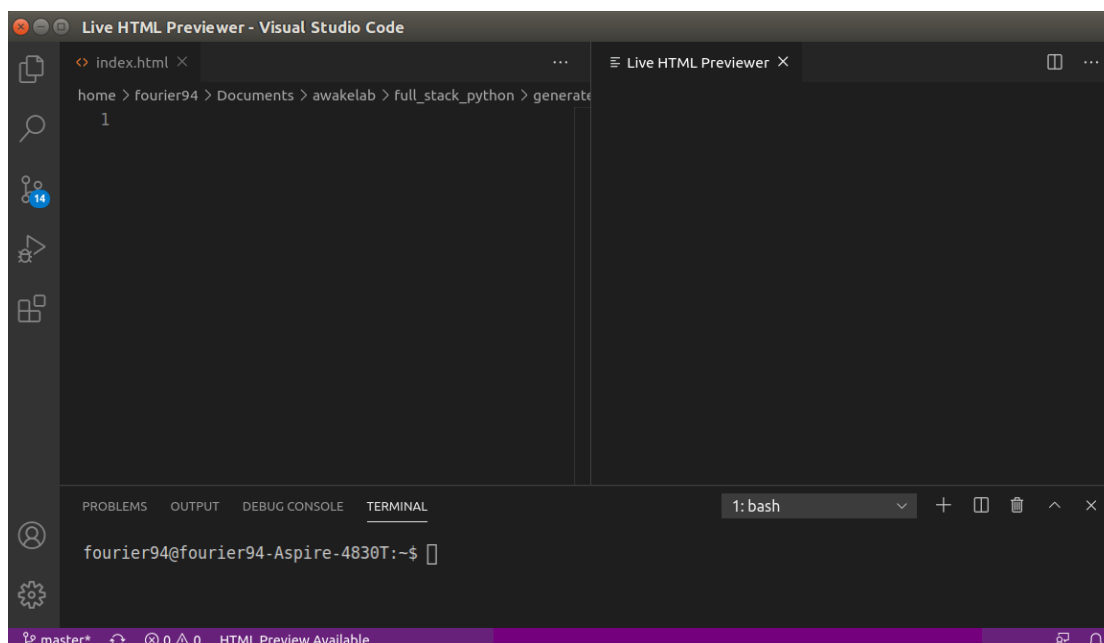
Una vez instalada la extensión, creamos un nuevo archivo de texto en **Archivo --> Nuevo Archivo** y lo guardamos con el nombre **index.html**.

Además, para ambientarnos a la forma en que trabajaremos en el resto del curso, abrimos una vista de terminal con **Terminal --> Nuevo Terminal**.

De esta forma tendremos algo como esto:



Para aprovechar las facilidades de nuestra nueva extensión de previsualización de HTML, dividiremos nuestra vista del editor en dos columnas. Manteniendo nuestro archivo creado `index.html` a la izquierda y pediremos al editor que al costado derecho despliegue la previsualización del documento HTML como se vería en un navegador. Para esto debemos presionar F1 y escribimos [Show HTML side preview](#), con lo que obtendremos nuestro layout final que será como el que se muestra a continuación:

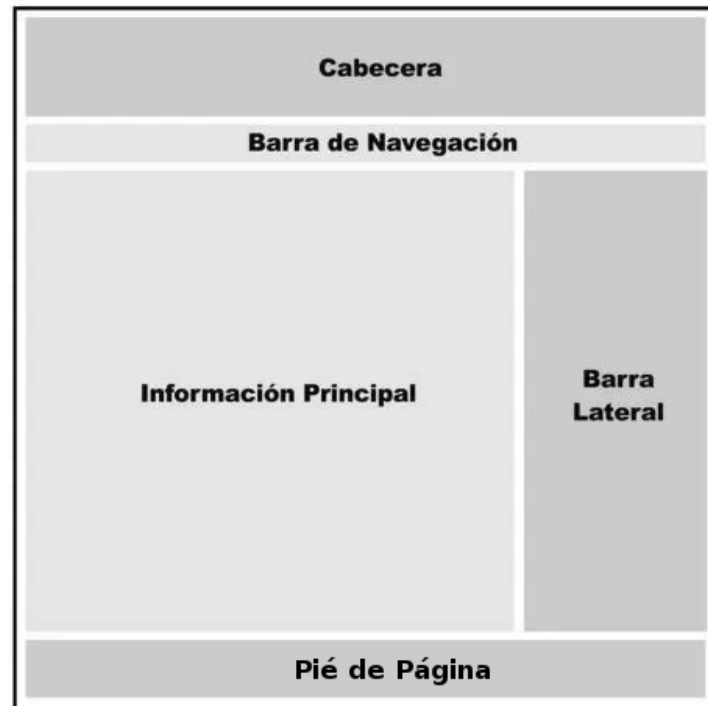


Según lo que hemos visto previamente, la estructura básica del documento HTML está compuesto por **<doctype>**, **<html>**, **<head>** y **<body>**. Luego llenaremos con otros elementos interiores esta estructura básica de inicio. De esta forma comenzamos a incluir los primeros elementos de nuestro código HTML en la ventana del costado izquierdo:

```
1  <!DOCTYPE html>
2  <html lang="es">
3    <head>
4
5    </head>
6    <body>
7
8    </body>
9  </html>
```

Dentro de la sección **<head>** incluiremos 2 elementos básicos: **<title>**, para la definición del texto de título que aparecerá en la lengüeta superior del navegador; y un elemento **<meta>** con el que definiremos la codificación de caracteres de nuestra página.

Por otro lado, en la sección **<body>** emplazaremos todo el contenido que será visible para el usuario como parte del documento. Nos hacemos la idea de un sitio con la siguiente estructura:



Con esto seleccionamos desde la sección **1.1.1.6. Etiquetas HTML5** las etiquetas que nos permiten implementar la estructura mostrada.

De esta manera, volviendo a nuestro editor podemos modificar nuestro código de la siguiente forma:

Si experimentamos con el código retirando las etiquetas `<header>`, `<nav>`, `<section>`, `<article>`, `<aside>` y `<footer>`, veremos que no hay un efecto visual en este momento. Sin embargo, las utilizamos pues más adelante cuando deseemos personalizar características de visualización a nuestro sitio serán de utilidad. Además estas etiquetas semánticas permiten funcionalidades externas de nuestro sitio, como mejor comprensión de nuestro sitio por motores de búsqueda o indexación.

Archivo: [index.html](#)

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Página de Enlaces</title>
    <meta charset="utf-8"/>
  </head>
  <body>
    <header>
      <h1>Página de Enlaces</h1>
      <p>Actualizado: 20-06-2020</p>
    </header>
    <nav>
      <p>
        <a href="#noticias-destacadas">Noticias Destacadas</a> |
```



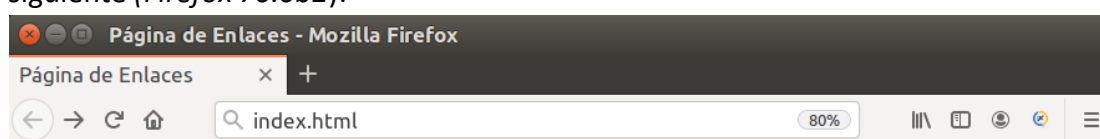
```
<a href="#motores-de-busqueda">Sitios de Búsqueda</a> |
<a href="#diarios">Diarios</a> |-
<a href="#finanzas">Finanzas</a>
</p>
</nav>
<section id="noticias-destacadas">
<h3>Noticias Destacadas del Día</h3>
<ul>
<li><p>[07-06-2020 15:22] Noticia 1.
<a href="#">ver más</a></p></li>
<li><p>[06-06-2020 16:03] Noticia 2.
<a href="#"> ver más</a></p></li>
</ul>
</section>
<section id="motores-de-busqueda">
<h3>Motores de Búsqueda</h3>
<ol>
<li><p>Buscar en <a href="https://google.com">
Google</a></p></li>
<li><p>Buscar en <a href="https://duckduckgo.com/">
Duck Duck Go</a></p></li>
</ol>
</section>
<section id="diarios">
<h3>Diarios Nacionales e Internacionales</h3>
<ul>
<li><p>Revisar noticias en
<a href="https://www.washingtonpost.com/">
Washington Post</a></p></li>
<li><p>Revisar noticias en <a href="https://www.emol.com">
Emol</a></p></li>
</ul>
</section>
<section id="finanzas">
<h3>Sitios de Finanzas</h3>
<ul>
<li><p>Ir a <a href="https://www.bolsadesantiago.com/">
Bolsa de Comercio de Santiago</a></p></li>
<li><p>Ir a <a href="https://es-us.finanzas.yahoo.com/">
Yahoo Finanzas</a></p></li>
</ul>
</section>
<br>
<aside >
Versión en Español
</aside>
<br>
<footer>
Copyright &copy; 2019-2020
</footer>
</body>
</html>
```

En la ventana de Preview de nuestra extensión en Visual Studio Code podremos ver el resultado visual de nuestra página a medida que vayamos escribiendo este código. Debemos ver algo como lo siguiente:



```
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Página de Enlaces</title>
5     <meta charset="utf-8"/>
6   </head>
7   <body>
8     <header>
9       <h1>Página de Enlaces</h1>
10      <p>Actualizado: 20-06-2020</p>
11    </header>
12    <nav>
13      <p>
14        <a href="#noticias-destacadas">Noticias Destacadas</a>
15        <a href="#motores-de-busqueda">Sitios de Búsqueda</a>
16        <a href="#diarios">Diarios</a>
17        <a href="#finanzas">Finanzas</a>
18      </p>
19    </nav>
20    <section id="noticias-destacadas">
21      <h3>Noticias Destacadas del Día</h3>
22      <ul>
23        <li><p>[07-06-2020 15:22] Noticia 1. <a href="#">ver más</a></p></li>
24        <li><p>[06-06-2020 16:03] Noticia 2. <a href="#">ver más</a></p></li>
25      </ul>
26    </section>
27    <section id="motores-de-busqueda">
28      <h3>Motores de Búsqueda</h3>
29      <ol>
```

Esta página no contiene ningún tipo de especificación de estilos, colores, bordes, espaciado u otros. Todos los elementos se muestran, por lo tanto, con el aspecto estándar que el navegador posee preconfigurado según su soporte del estándar HTML. Si abrimos nuestro archivo [index.html](#) en un navegador, veremos algo como lo siguiente (Firefox 70.0b2):



Página de Enlaces

Actualizado: 20-06-2020

[Noticias Destacadas](#) | [Sitios de Búsqueda](#) | [Diarios](#) | [Finanzas](#)

Noticias Destacadas del Día

- [07-06-2020 15:22] Noticia 1. [ver más](#)
- [06-06-2020 16:03] Noticia 2. [ver más](#)

Motores de Búsqueda

1. Buscar en [Google](#)
2. Buscar en [Duck Duck Go](#)

Diarios Nacionales e Internacionales

- Revisar noticias en [Washington Post](#)
- Revisar noticias en [Emol](#)

Sitios de Finanzas

- Ir a [Bolsa de Comercio de Santiago](#)
- Ir a [Yahoo Finanzas](#)

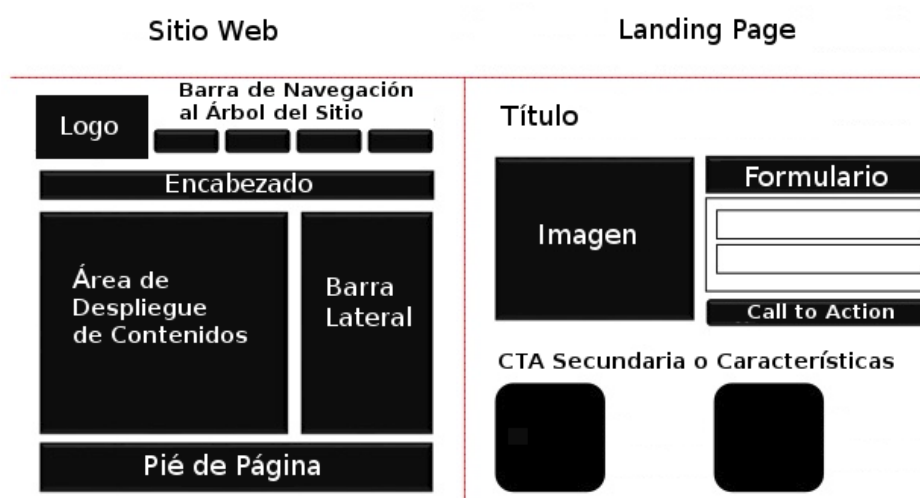
Versión en Español

Copyright © 2019-2020



1.1.1.8. Landing pages vs sitios web

Entre los recursos o aplicaciones HTML que se pueden encontrar se destacan los Sitios Web, la Landing Pages y cualquier software de mayor complejidad basado en tecnologías Web; tales como sistemas de gestión interna de empresas, manejo de clientes, transacciones financieras, etc. Algunas de estas aplicaciones están abiertamente disponibles por internet, otras requieren autenticación para acceder a una parte de sus contenidos, y otras sólo se utilizan en redes internas de empresas para propósitos privados.



Vistas típicas de un Sitio Web y una Landing Page

Si nos centramos en las categorías de Sitios Webs y Landing Pages, veremos que ambas aplicaciones son para propósitos distintos:

SITIOS WEB

Los Sitios Web corresponden típicamente a un conjunto integrado de páginas ordenadas y conectadas por links según una estructura de árbol que permite orientar y guiar al usuario a acceder a información relacionada a organizaciones, empresas, académicas, personales, productos o servicios, en las que los visitantes pueden leer, consultar, comprar o interactuar de cualquier forma.

Un Sitio Web a menudo posee enlaces en la parte superior y al costado de las páginas para contenidos como: Acerca de, Servicios, Información de la empresa u organización, Blog, etc. (Básicamente, las secciones que son clave para su negocio). También pueden tener imágenes o enlaces que conducen a páginas externas al sitio.

LANDING PAGE

Las Landing Pages son una forma de página web que, por lo general, están destinadas a un propósito muy específico; como registrarse, recopilar información



del visitante como potencial cliente o seguidor, o vender un producto. La principal aplicación es la generación de tráfico para un objetivo específico de campaña de marketing y recolectar información de contacto de los visitantes para realizar acciones de marketing posterior. La intención es enfocar al visitante únicamente en la intención de la página, como el proceso de registro. La diferencia entre una landing page y una web, consiste en la conversión ya que el usuario no se dispersa, recibe la información que necesita y si esta interesado dejará sus datos.

Durante mucho tiempo la opciones de conversión dentro de una landing page se han focalizado en conseguir; a traves de formularios; números telefónicos, emails o perfiles de redes sociales. Esos eran los objetivos con perseguían los especialistas en marketing. Sin embargo, el consumidor ha evolucionado y cada vez quiere más otro tipo de interacciones en tiempo real sin esperas. De esta forma los chatbots han aparecido como una herramienta nueva para generar interacciones en tiempo real, 24/7 con respuestas rápidas y conseguir más leads, y aumentar la tasa de conversión de usuarios visitantes a usuarios que ejecutan la acción.

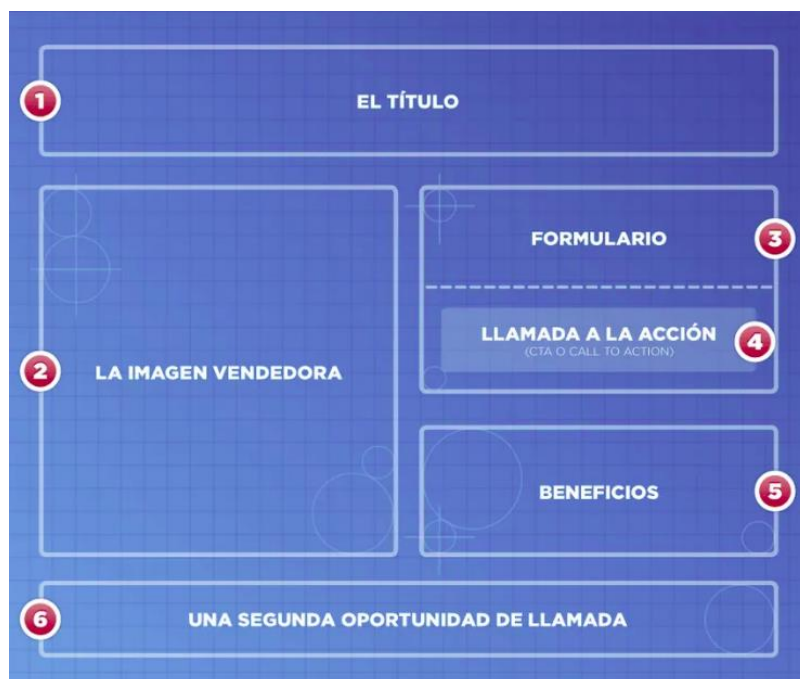
Las landing page no se crean para posicionarse en google, sino para atraer de la forma más rápida al mayor número de leads. Las estrategias para atraer clientes potenciales a una landing page, pueden ser:

- Anuncios patrocinados o de pago por click (PPC) como Google Adwords o Facebook Ads.
- Mediante el uso de las redes sociales, ejemplo Facebook y Twitter.
- Mediante un código QR, como el uso del QR en un flyer.

Una vez tengamos una base de datos óptima y optimizada será el momento de crear lo que es conocido como embudo de marketing, que consiste en programar una serie de emails informativos y de venta, para informar y convertir. En este sentido habrá que ir depurando qué emails se abren más y cuáles tienen una mayor conversión^{[14][15][16]}.

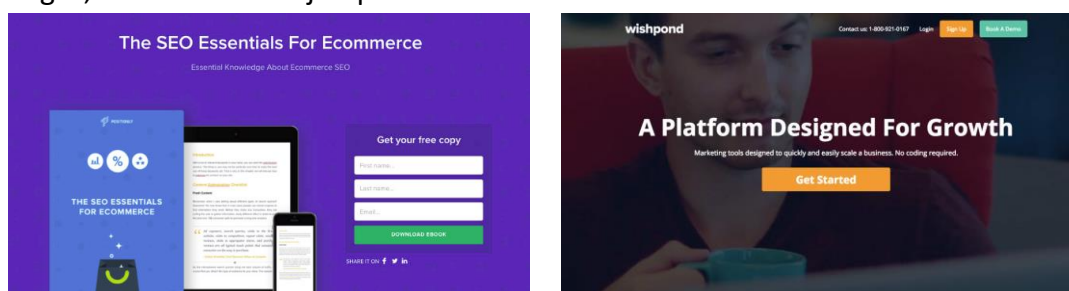
1.1.1.9. Estructura de una landing page

Para los fines descritos en la sección anterior, implementaremos una Landing Page básica, consideraremos la estructura esbozada en la imagen anterior, con las secciones: 1) Título, 2) Imagen, 3) Formulario, 4) Call to Action, 5) Beneficios y 6) Segunda oportunidad de Call to Action; tal como se detalla en la siguiente imagen:



Anatomía de una Landing Page^[16]

Revisando la world wide web podemos encontrarnos con variedades de Landing Pages, tales como los ejemplos mostrados a continuación:



Ejemplos de layouts de Landing Pages^[17]

El conocimiento que hemos acumulado en hasta el momento, nos da la posibilidad de crear la vista HTML de nuestra propia Landing Page, donde incluiremos las secciones más arriba mencionadas.

La contenido **1) Título** la implementamos dentro del elemento **<header>**, a través de **<h1>** y **<h2>**, elementos que confirmar un título y subtítulo. Creamos además el elemento **<section id="imagen">** en el cual emplazamos el contenido **2) Imagen**. Hemos introducido aquí el concepto de **id** de un elemento del documento HTML. A través de este atributo podemos asignar un identificador único del elemento, que en este caso hemos llamado arbitrariamente "imagen", y a través del cual podemos hacer referencia a éste. Los contenidos **3) Formulario**, **4) Call to Action** y **5)**



Beneficios las ubicamos dentro de un elemento `<section id="formulario-beneficios">`. Introducimos también en este ejemplo el uso de un elemento `<style>`, que define ciertas características visuales que deseamos en nuestros elementos HTML. Principalmente logramos habilitar el despliegue en dos columnas distintas para las secciones “imagen” y “formulario-beneficios”, a las que le asignamos el 60% y 40% del ancho de la página, respectivamente. Nótese que en las definiciones de estilo el estandar hace referencia a los atributos **id** a través del símbolo **#**. En la misma sección de estilos se puede observar que los elementos identificados por el nombre de la etiqueta HTML se referencian sólo con el nombre de etiqueta.

El código completo que escribimos en nuestro editor de texto, Visual Studio Code, es como sigue:

Archivo: landing_page.html

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Landing Page</title>
    <meta charset="utf-8"/>
    <style>
      #imagen {
        float:left;
        width:60%;
        text-align: center;
      }
      #formulario-beneficios {
        float:right;
        width:40%;
      }
      header, footer {
        clear:both;
        text-align: center;
      }
    </style>
  </head>
  <body>
    <header>
      <h1>Nuestro Producto Estrella</h1>
      <h2>Regístrate dentro de los 100 primeros
        y recibirás un descuento</h2>
    </header>
    <section id="imagen">
      
      <p>Image copyrights to https://www.cfg.com/</p>
    </section>
    <section id="formulario-beneficios">
      <form>
        <p>Nombre</p>
        <input type="text">
        <p>Email</p>
        <input type="email"/>
        <br>
        <br>
        <button type="submit">Regístrate</button>
      </form>
    </section>
  </body>
</html>
```



```
</form>
<br>
<h3>¡Nuestros Beneficios!</h3>
<ul>
  <li>Rapidez</li>
  <li>Seguridad</li>
  <li>Gran Precio</li>
  <li>Flexibilidad</li>
</ul>
</section>
<footer>
<br>
<p>
  <a href="">Like Red Social 1</a> |
  <a href="">Like Red Social 2</a> |
  <a href="">Like Red Social 3</a> |
  <a href="">Like Red Social 4</a>
</p>
<br>
<p>Copyright &copy; 2019-2020 -
  Desarrollos de Landing Pages Inc.</p>
</footer>
</body>
</html>
```

En la ventana de Preview de nuestra extensión en Visual Studio Code podremos ver el resultado visual de nuestra página a medida que vayamos escribiendo este código. Debemos ver algo como lo siguiente:



Nuestro Producto Estrella

Regístrate dentro de los 100 primeros y recibirás un descuento



Image copyrights to <https://www.cfg.com/>

Nombre

Email

Regístrate

¡Nuestros Beneficios!

- Rapidez
- Seguridad
- Gran Precio
- Flexibilidad

[Like Red Social 1](#) | [Like Red Social 2](#) | [Like Red Social 3](#) | [Like Red Social 4](#)

Copyright © 2019-2020 - Desarrollos de Landing Pages Inc.



1.1.1.10. Navegador, DOM, Inspector de elementos.

El Navegador Web es una de las herramientas digitales más conocidas y utilizadas a nivel de usuario en las últimas décadas. Un usuario básico de la web será agnóstico al navegador a través del cual acceder a los recursos que se encuentran en la World Wide Web.

Ya explicamos anteriormente que el Navegador Web es un caso particular de agente HTTP. Cuando el navegador web obtiene datos de un servidor conectado a Internet, utiliza un software interno llamado motor de renderizado para traducir esos datos en texto e imágenes. Estos datos están escritos en lenguaje de marcado de hipertexto (HTML) y los navegadores web leen este código para crear lo que vemos, escuchamos y experimentamos en Internet.

Los Navegadores Web generalmente son grandes proyectos de software desarrollado por muchos colaboradores del mundo del software libre o de compañías de desarrollo. Éstos compiten por ofrecer la mejor herramienta que balancee los intereses de usuarios con los de empresas que muchas veces impulsan características que crean ventajas competitivas y mejor soporte para las aplicaciones web que ofrecen al mundo a través de internet.

La velocidad y la privacidad son dos de las principales consideraciones que lo guiarán cuando elija un navegador; algunos demandan más recursos de su sistema, mientras que otros son relativamente livianos, y algunos ofrecen conjuntos completos de herramientas de seguridad para proteger su identidad en línea, mientras que otros permiten que las cookies y los anuncios se ejecuten sin obstáculos.

A nivel de usuario, al acceder a sitios webs o páginas nos damos cuenta que, en algunos casos, hay diferencias en la manera de desplegar la información de texto, estilos, imágenes o videos. Esto se debe a que los navegadores, como proyectos de software constantemente en desarrollo, actualizan dinámicamente su soporte de versiones de los distintos estándares de tecnologías web que hemos revisado en secciones anteriores (HTTP, HTML, CSS, Javascript, etc.). Esto hace que muchas veces estos desarrollos vayan desfasados en el tiempo y algunos de los navegadores soporten más tempranamente que otros algunas características.

De esta forma, desde el punto de vista del desarrollador, es importante tener presente que durante el desarrollo es importante tener acceso a distintos navegadores y versiones de éstos, además de distintas plataformas de hardware (Tipos de PC, Tablets y Smartphones). En caso de aplicaciones Web intra empresariales, muchas veces se escoge maximizar el grado de compatibilidad con un navegador en particular, y se establece como requisito la utilización de éste como una regulación a los trabajadores de la compañía. Sin embargo, en aplicaciones abiertamente disponibles en la web, es mandatorio el poder asegurar una operación correcta en la mayoría de los navegadores del mercado y así llegar adecuadamente a



la mayor cantidad de usuarios independiente del dispositivo, sistema operativo y navegador que posean.

Los navegadores que comparten la mayor parte del mercado en la actualidad son los conocidos: **Mozilla Firefox**, **Google Chrome**, **Microsoft Edge** and **Apple Safari**. Existen otros navegadores de menor adopción tales como Opera, Vivaldi, Brave, Torch, Midori, etc. Es interesante destacar que actualmente muchos de los navegadores disponibles están basados en el proyecto opensource de Google llamado **Chromium**, tales como Chrome, Vivaldi, Torch, Brave y recientemente Microsoft migró su navegador a una versión basada en Chromium. Aunque Chromium es un proyecto opensource de Google, esta compañía desarrolla la versión Chrome con características propietarias, como actualizaciones o soporte de funcionalidades específicas de sus suites de aplicaciones^{[18][19]}.

La distribución de mercado entre los principales navegadores en la actualidad (Mayo 2020) son:

Cuota de Mercado Desktop/Laptop			Cuota de Mercado Móvil		
Chrome	Safari	Firefox	Chrome	Safari	Samsung Internet
68.33%	9.4%	8.91%	61.61%	24.45%	6.51%
Edge Legacy	IE	Opera	UC Browser	Opera	Firefox
4.41%	3%	2.41%	3.79%	1.55%	0.43%

Fuente: <https://gs.statcounter.com/>

En particular Mozilla, se ha comprometido a respaldar la privacidad de sus usuarios y desarrollar herramientas para evitar que terceros lo rastreen en la web. En la gran diversidad de navegadores, otros están avocados a asegurar la privacidad como una de sus principales preocupaciones, entre estos se encuentran: Iridium, Tor, GNU IceCat^[20].

El navegador, a parte de ser la herramienta de acceso a la web por usuarios de todo el mundo, es una herramienta esencial para todo desarrollador. Las versiones estandar de los navegadores típicos poseen un set de herramientas para desarrolladores. Si revisamos Chrome y Firefox encontraremos un set de herramientas estructuradas de forma muy similar. Existen además versiones para desarrolladores, de Chrome y Firefox, que vienen configurados con opciones que facilitan las operaciones de desarrollo y además están más avanzados (Por ejemplo 3 meses) en términos de inclusión de funcionalidades comparados con la versión estandar. En lo que sigue utilizaremos principalmente Firefox Developer Edition para trabajar con herramientas de desarrollador en ambiente de navegador.

Para ilustrar en primera aproximación las facilidades de las herramientas de desarrollador, cargaremos en Firefox la landing page que desarrollamos en la sección



1.1.1.9 anterior. Al acceder a Menu --> Desarrollador Web --> Inspector podremos ver que se abre el set de herramientas de desarrollo del navegador:

Vista de Navegación



Vista de Desarrollador



En esta función de inspector, podemos observar que en la parte inferior de la vista de desarrollador tenemos acceso al código HTML de la página. Al hacer clic sobre alguno de los elementos, éste se resalta en la zona de despliegue, y en la parte inferior derecha podemos ver detalles de estilo del elemento seleccionado. Esta herramienta de inspección nos permite experimentar visualmente alterando código y estilos de nuestra página en desarrollo y de esta forma llegar a configuraciones que optimicen el resultado que deseamos obtener.

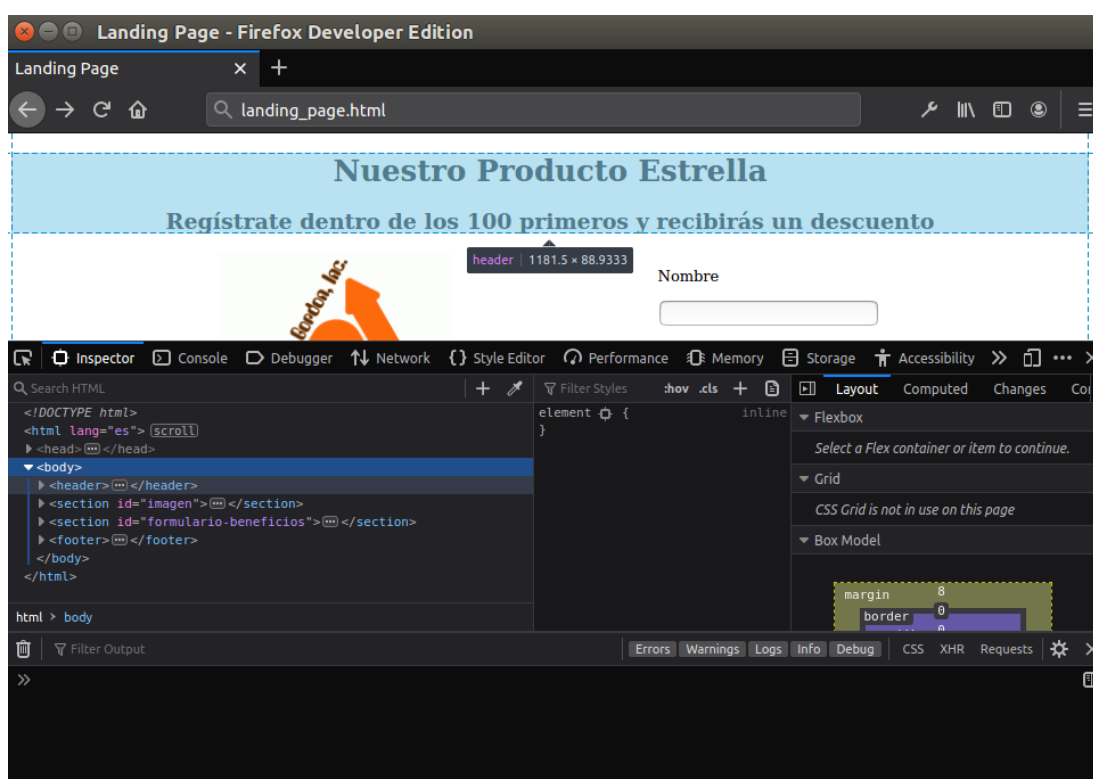
Además de Inspector, existen otras herramientas, entre las que se cuentan^[21]:

- **Consola:** Muestra los logs de la ejecución de la página web: cualquier solicitud de red, JavaScript, CSS, errores de seguridad y advertencias, así como también, advertencias y mensajes informativos explícitamente generados por Javascript en tiempo de ejecución dentro del contexto de la página. Permite interactuar con la página ejecutando expresiones Javascript en el contexto de la página.
- **Depurador (Debugger):** Permite depurar nuestro código, insertando puntos de detención o quiebre de nuestros scripts o ejecutar paso a paso secuencialmente las instrucciones que hemos incorporado en el código javascript de nuestra página.



- **Red:** El monitor de red muestra todas las solicitudes de red que Firefox realiza (por ejemplo, cuando carga una página, o debido a XMLHttpRequests), cuanto tiempo toma cada petición, y los detalles de cada petición.
- **Rendimiento:** Es una herramienta técnica que entrega una visión general de la capacidad de respuesta del sitio que visitas así como de su código JavaScript. Se pueden realizar registros de todas las acciones que el navegador empleó para generar el sitio web así como una gráfica con el tiempo de respuesta de dicha pagina.
- **DOM:** Permite inspeccionar las propiedades del DOM como una estructura de árbol expandible, comenzando desde el objeto raíz Window de la página actual o de un iframe seleccionado.

A continuación mostramos una vista útil de las herramientas de desarrollador, ubicándonos en Inspector, y agregando un panel inferior para ver la consola javascript, además de los códigos de estilo, todo en una misma vista:



Escogiendo la herramienta DOM, podemos tener acceso a todos los objetos de la API de los elementos del documento HTML actualmente cargado, en este caso de nuestra landing page:



En el transcurso de las siguientes secciones y módulos haremos uso más específico de algunas de estas funcionalidades.

1.1.2.- Organización de los assets de un proyecto web

Hasta el momento hemos generado código que principalmente incluye elementos HTML y hemos incursionado de forma tangencial en algunos conceptos de estilo y de scripting en las secciones 1.1.1.3 y 1.1.1.9. Hemos incluido algunos códigos CSS y Javascript básicos dentro del mismo documento. Por otro lado, sólo hemos hecho uso de código propio y no hemos recurrido a ningún módulo externo preexistente, ni para estilos ni para lógica programática en javascript. Sólo hemos llamado una imagen externa que fue desplegada en nuestra landing page.

En un proyecto real, lo más usual es que separemos del código HTML los bloques que pertenecen a CSS y Javascript. Esto es debido a la cantidad de líneas de los archivos de texto y para hacer más expedita su lectura e intervención, así como para facilitar la eventual división de trabajo entre distintos colaboradores que estén desarrollando el mismo proyecto. De la misma forma, si deseamos incluir en nuestro desarrollo otras hojas de estilo CSS o código Javascript preexistentes (Librerías), desarrolladas por nosotros mismos u otros desarrolladores, será apropiado que estén ubicadas como un archivo externo a nuestro código HTML principal. Los



archivos de imágenes, logos, datos, y cualquier otro archivo estático que sea un recurso para nuestra página o sitio, la ubicaremos en una posición que provea un orden y entendimiento claro dentro del proyecto.

Así, si exploramos en la web diversos proyectos HTML, en sitios estáticos disponibles en servidores, o si observamos la estructura de plantillas de sitios web que se entregan libremente o de pago, veremos que hay un consenso de buenas prácticas para estructurar los distintos archivos del proyecto.

EL DIRECTORIO DEL PROYECTO

Una estructura típica de archivos y directorios de un proyecto web es como la que se muestra a continuación:

```
proyecto/
├── css/
│   ├── style.css
│   ├── bootstrap.css
│   └── normalize.css
├── js/
│   ├── main.js
│   ├── jquery.js
│   └── bootstrap.js
├── img/
│   ├── logo.png
│   ├── product1.jpg
│   ├── product2.jpg
│   ├── photo1.jpg
│   └── photo2.jpg
├── common/
│   ├── article.html
│   ├── footer.html
│   └── navbar.html
├── fonts/
│   ├── Lato-Bold.ttf
│   ├── Lato-Black.ttf
│   └── Lato-Regular.ttf
├── index.html
├── empresa.html
├── productos.html
└── contacto.html
```

El directorio raíz **proyecto** contiene todos los directorios y archivos html principales de nuestro proyecto.

Los directorios **css** y **js** contienen todos los archivos de hojas de estilo y códigos javascript, respectivamente. En ambos casos podrán existir archivos que hayamos



desarrollado específicamente para el proyecto y otros que correspondan a librerías externas que estemos incorporando para potenciar nuestro desarrollo.

En el directorio **img** situaremos los archivos de imágenes que sean parte de las páginas que estemos desarrollando.

El directorio **common** se utiliza algunas veces, en casos en que existen bloques de código HTML idéntico que es utilizado en múltiples lugares o archivos de nuestro proyecto. En estos casos, se incluye código javascript, que permite al browser incluirlos en el instante de cargar la página respectiva, y de esta forma ahorrarnos la repetición redundante de esos bloques en las distintas páginas; por ejemplo barras de navegación, piés de página, etc. Por ejemplo, para incluir una barra de navegación que se repite en todas nuestras páginas bastaría con crear sólo un elemento **<div>** que fuera poblado al momento de la carga con el siguiente código javascript:

```
<div id="barra-nav"></div>
<script>
  $(function(){
    $("#barra-nav").load("common/navbar.html");
  });
</script>
```

El directorio **fonts** se utiliza muchas veces cuando se desea utilizar cierta tipografía especial que probablemente no esté disponible como recurso estandar en el browser. De esta manera podemos asegurar que el aspecto de nuestro diseño se mantenga.

Esta estructura puede ser más simple o más sofisticada. Por ejemplo, podría requerirse la creación de subdirectorios para dividir archivos css o js, propios o librerías. En el caso de imágenes, que pueden estar en gran cantidad, podríamos dividirla por directorios correspondientes a cada una de las páginas del proyecto.

FORMATO DE ARCHIVOS DE IMAGEN

Existe una gran variedad de formatos de imágenes disponibles que son soportadas por los distintos navegadores web del mercado.

Abreviación	Formato	MIME type	Extensión	Compatibilidad
APNG	Animated Portable Network Graphics	image/apng	.apng	Chrome, Edge, Firefox, Opera, Safari
BMP	Bitmap file	image/bmp	.bmp	Chrome, Edge, Firefox,



				Internet Explorer, Opera, Safari
GIF	Graphics Interchange Format	image/gif	.gif	Chrome, Edge, Firefox, Internet Explorer, Opera, Safari
ICO	Microsoft Icon	image/x-icon	.ico, .cur	Chrome, Edge, Firefox, Internet Explorer, Opera, Safari
JPEG	Joint Photographic Expert Group image	image/jpeg	.jpg, .jpeg, .jif, .pjpeg, .jpg	Chrome, Edge, Firefox, Internet Explorer, Opera, Safari
PNG	Portable Network Graphics	image/png	.png	Chrome, Edge, Firefox, Internet Explorer, Opera, Safari
SVG	Scalable Vector Graphics	image/svg+xml	.svg	Chrome, Edge, Firefox, Internet Explorer, Opera, Safari
TIFF	Tagged Image File Format	image/tiff	.tif, .tiff	None built-in; add-ons required
WebP	Web Picture format	image/webp	.webp	Chrome, Edge, Firefox, Opera

GIF es una buena opción para imágenes y animaciones simples, aunque la conversión de imágenes en color a GIF puede ser insatisfactoria. Por lo general, el contenido moderno debe usar **PNG** (Se pronuncia “pint”) para imágenes fijas sin pérdida e indexadas, y debe considerar usar APNG para secuencias de animación sin pérdida.

JPEG is currently the most widely used lossy compression format for still images. It's particularly useful for photographs; applying lossy compression to content requiring sharpness, like diagrams or charts, can produce unsatisfactory results.

Los archivos SVG son archivos de texto que contienen código fuente que, cuando se interpretan, dibujan la imagen deseada. Este tipo de imagen vectorial se utiliza en ámbitos como mapas, dibujos de ingeniería, entre otros.

WebP soporta compresión con pérdida a través de la codificación predictiva basada en el códec de video VP8 y compresión sin pérdida que utiliza sustituciones para datos repetitivos. Las imágenes con pérdida de WebP son de 25 a 35% más pequeño que las imágenes JPEG de niveles de compresión visualmente similares. Las imágenes WebP sin pérdida son típicamente un 26% más pequeñas que las mismas imágenes en formato PNG^[22].



CDN (CONTENT DELIVERY NETWORK)

Las CDN son un grupo de servidores distribuidos geográficamente. Estos servidores almacenan copias duplicadas de datos para que servidores de todo el mundo puedan cumplir con los requests HTTP de datos para los usuarios desde la ubicación más cercana.

En la sección “EL DIRECTORIO DEL PROYECTO” mencionamos la residencia de múltiples archivos css o javascript en distintos directorios del proyecto. En caso de archivos correspondientes a librerías estándar, como por ejemplo jQuery, Bootstrap u otros, es posible entregar una mejor experiencia al usuario cargando estos archivos desde servidores más cercanos incluso que nuestro servidor de aplicación o, en último, caso descargarlo de esta tarea que puede ser cumplida por otros.

Como mencionamos, las CDNs se usan para proveer hojas de estilo y archivos Javascript (activos estáticos) de bibliotecas como Bootstrap, jQuery, etc. Es preferible usar CDN para esos archivos de biblioteca por varias razones^[23]:

RUTAS RELATIVAS Y ABSOLUTAS

En nuestros archivos HTML frecuentemente deberemos hacer referencia a recursos, archivos, que se encuentran en alguna posición de nuestra estructura de archivos del proyecto. De la misma forma, es posible que estemos haciendo referencia a recursos que estén en otros servidores de la web. De esta forma surgen dos tipos de referencias, las basadas en Rutas Relativas o en Rutas Locales.

En un proyecto web, el directorio raíz estará normalmente asignado a la raíz del dominio que hayamos definido para éste. Por ejemplo la dirección: <http://www.nuestralandingpage.cl/> corresponderá a la raíz de nuestro proyecto.

De esta manera, en elementos HTML dónde debamos apuntar, por ejemplo, a un archivo de imagen almacenado localmente en nuestro servidor, lo más evidente sería hacerlo de la siguiente forma:

```

```

Sin embargo, este tipo de referencia, a la cual llamamos **Ruta Absoluta** al archivo, tiene inconvenientes, tales como el dificultar un eventual cambio de dominio donde estemos desplegando nuestro proyecto. Tal caso requeriría que se buscaran todas las líneas de código donde existe algún enlace a recursos internos de nuestro servidor para hacer la actualización que corresponda. Alternativamente, una práctica más adecuada es utilizar lo que llamamos **Ruta Relativa** al archivo, donde toda



referencia a un recurso del mismo servidor está en relación al directorio raíz de nuestro proyecto. Por lo tanto la URI <http://www.nuestralandingpage.cl/> estará considerada implícitamente dentro de cualquier link y sólo indicaremos la ruta al recurso deseado, desde esa posición. Por ejemplo, en nuestro caso, indicaremos el enlace al archivo de imagen de la siguiente manera:

```

```

Por último, pensemos en un caso en que todos los archivos HTML de las distintas secciones de nuestro sitio web, que no correspondan a [index.html](#), los hayamos ubicado en un directorio llamado **secciones**. En este caso los archivos de imágenes estarán ubicados en un directorio una nivel más atrás del actual. En ese caso debemos utilizar la sintaxis `../` para indicar esta situación. De esta forma los enlaces desde nuestro archivo HTML quedarían de la siguiente forma:

```

```

Las **Rutas Absolutas** son necesarias en casos que introduzcamos enlaces a recursos ubicados en otros servidores de la web, pues en esos casos no hay otra forma que indicar explícitamente la ruta para encontrarlos. Por ejemplo, en el caso de recursos a obtener desde un CDN, como archivos css o javascript, deberemos utilizar rutas absolutas como estas:

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.cs
s">

<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
```

Preferiremos utilizar **Rutas Relativas** para todos los recursos que residen en nuestro propio servidor.

Cuando veamos el desarrollo de sitios web dinámicos podremos ver que la problemática de rutas relativas generalmente la maneja el framework de desarrollo web que utilicemos, que en nuestro caso será basado en Python y llamado Django.



1.1.3.- Referencias

- [1] The Manchester Small Scale Experimental Machine -- "The Baby"
<http://curation.cs.manchester.ac.uk/computer50/www.computer50.org/index.html>
- [2] Breve historia de internet
<https://www.internetsociety.org/es/internet/history-internet/brief-history-internet/>
- [3] History of the Web
<https://webfoundation.org/about/vision/history-of-the-web/>
- [4] Definition of User Agent
https://www.w3.org/WAI/UA/work/wiki/Definition_of_User_Agent
- [5] HTTP: The Definitive Guide, O'Reilly & Associates, Inc.
- [6] Introduction to the server side
https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction
- [7] J.D Gauchat "HTML5 para Mentas Maestras" 2a edición
- [8] HTML Living Standard, by WHATWG.org
<https://html.spec.whatwg.org/multipage/>
- [9] Difference between Node object and Element object.
<https://stackoverflow.com/questions/9979172/difference-between-node-object-and-element-object>
- [10] Introduction to the DOM, by mozilla.org
https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
- [11] What is the difference between an IDE and an editor?
<https://discuss.atom.io/t/what-is-the-difference-between-an-ide-and-an-editor/32629>
- [12] Text Editors and Word Processors
<https://idrh.ku.edu/text-editors-and-word-processors>
- [13] Lista de Elementos HTML5 , by mozilla.org
https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5_lista_elementos
- [14] Landing Pages vs Web Pages: What's the Difference?



<https://www.digitalhill.com/blog/landing-pages-vs-web-pages-whats-difference/>

[15] Diferencias entre una web, un microsite y una landing page

<https://bilnea.com/diferencias-entre-web-microsite-landing-page/>

[16] Infografía: Anatomía de una Landing Page Perfecta

<https://www.bluecaribu.com/infografia-anatomia-de-una-landing-page-perfecta>

[17] The Perfect Landing Page Structure for SEO (and Conversions!)

<https://blog.wishpond.com/post/115675437245/landing-page-seo>

[18] The best browser 2020

<https://www.techradar.com/best/browser>

[19] What is a web browser?

<https://www.mozilla.org/en-US/firefox/browsers/what-is-a-browser/>

[20] Private and Secure Browsers

<https://restoreprivacy.com/secure-browser/>

[21] Firefox Developer Tools

<https://developer.mozilla.org/en-US/docs/Tools>

[22] Image file type and format guide

https://developer.mozilla.org/es/docs/Web/Media/Formats/Image_types

[23] CDN

<https://developer.mozilla.org/en-US/docs/Glossary/CDN>