

Unidad 3

Introducción a las aplicaciones Web y HTML5



Universidad Tecnológica Nacional
Facultad Regional Córdoba
Cátedra: Programación de Aplicaciones Visuales 2

Arquitectura de n-capas de una aplicación web



La Arquitectura de N-capas es probablemente uno de los modelos más utilizados en programación. Se utiliza tan a menudo porque es escalable, extensible, segura, fácil de mantener en el tiempo, reutilizable y se puede trabajar por diferentes programadores según su especialidad sin interferirse el trabajo.

Para el desarrollo de una arquitectura en capas robusta y perdurable en el tiempo, se deben tener en cuenta los siguientes aspectos:

- Las capas de una aplicación deben ser independientes entre sí, como las fichas de un lego. De esta manera podemos reemplazar cualquiera de las capas por una más actualizada en cualquier momento.
- Se debe mantener el principio de responsabilidad modular, esto quiere decir que cada capa es responsable de un tema o característica.
- Principio de caja negra. Cada componente es independiente de otros y solo interactúa en sus entradas y salidas.
- Don't Repeat Yourself (DRY). Principio de No repitas. Cada funcionalidad debe estar una sola vez en el sistema (Sin duplicados).
- Establecer normas en la codificación del desarrollo. Esto asegura la consistencia de código y el mantenimiento del mismo.

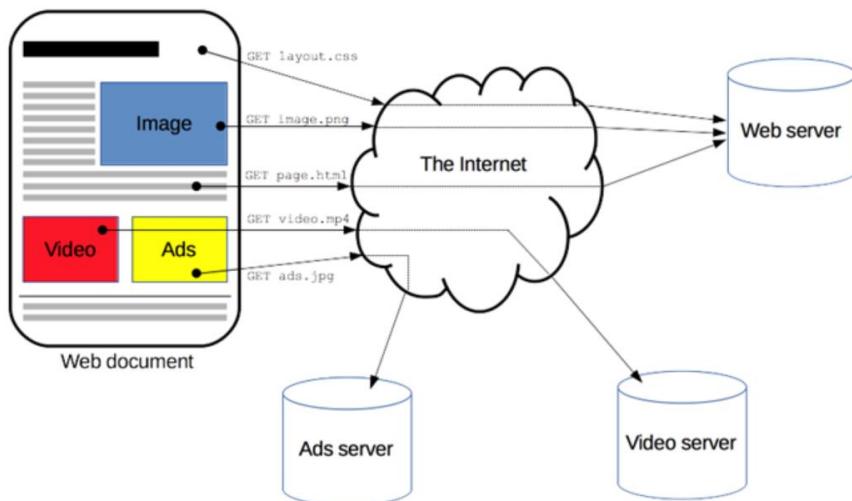
Ventajas

Siendo la programación en capas una arquitectura que tiene como objetivo principal separar la capa de diseño de la de lógica, existen otros valores adicionales que

aportan a proyectos grandes entre otros como:

- Facilita la creación de código estructurado
- Facilita el trabajo en equipo, permitiendo que varios programadores trabajen paralelamente en una misma funcionalidad sin interferirse mutuamente.
- Permite separar algunas de las capas de código en diferentes servidores para balanceo de cargas, buses de datos empresariales, etc.

¿Qué es una aplicación web SPA?



SPA, un paradigma de Arquitectura de Aplicaciones Web en auge

Dentro del desarrollo de aplicaciones web hay una tendencia importante a las denominadas SPA, Single Page Apps. Uno de los principales objetivos es conseguir una mejora importante en la experiencia de usuario: La mejora de los tiempos de espera o latencia entre vistas (similar a las aplicaciones nativas).

En esencia los motivos de las ventajas de las SPAs son sencillos, aunque creo que conviene ponerlo un poco en contexto.

Un SPA se carga completo durante el cargado de la página inicial y luego las regiones se reemplazan o se actualizan con los fragmentos de las nuevas páginas según petición del servidor. Para evitar descargas excesivas de características inutilizadas, un SPA descarga progresivamente las características cuando se necesiten, pueden ser fragmentos de las páginas o módulos completos de la pantalla.

De esta forma existe una analogía entre los "estados" de un SPA y las "páginas" de un sitio web tradicional. Como la navegación de estados en la misma página es análogo a la navegación de las páginas, en teoría, cualquier página de sitio web podría convertirse a un sitio de página única reemplazando las páginas solamente en las partes donde se generen un cambio. El enfoque de SPA en la página web es similar a Single Document Interface (SDI) que es una técnica para las aplicaciones de escritorio.

¿Qué diferencia hay respecto a aplicaciones web clásicas?

Habitualmente la lógica de negocio (el código ejecutable) de aplicaciones web se realiza íntegramente en el lado del servidor, y se confía la propia naturaleza del sistema de URLs el mostrar una “vista de aplicación” u otra. Por ej. en un e-commerce el carrito estará en una URL concreta, mientras que la pantalla de login estará bajo otra URL totalmente diferente. Para el navegador, cada URL diferente es completamente independiente del resto: Aunque tenga los mismos estilos y/o plantillas, estos tienen que volver a ser procesados desde cero. Esto, para la gran mayoría de páginas web dinámicas, implica que al cambiar entre vistas se sufrirá el problema de la latencia en la web (artículo de Ilya Grigorik, developer advocate de Google).

Otra característica negativa de esta arquitectura es que el estado de la aplicación del cliente es difícil de mantener, teniendo que hacer auténticos malabarismos para poder gestionar una simple transferencia de información de una vista a otra: Bien lo sabe aquel programador que haya tenido que implementar el típico asistente que se compone de diversas vistas, por ejemplo.

¿Podríamos encontrar más pegas? Puede, pero no es el objetivo. Es más, el sistema REST y URLs tiene mucho sentido y aporta muchas otras ventajas desde la misma arquitectura de servicios web (no interficies) hasta el facilitar el lado humano de compartir un recurso vía un enlace, aunque en ciertos aspectos juegue en nuestra contra.

Arquitectura básica de una SPA

En esencia una SPA es la interfaz de la aplicación web implementada casi íntegramente en el navegador (en lenguaje Javascript actualmente), aunque como toda página web tenga una base importante de HTML y CSS.

Todas las vistas de la interfaz de la aplicación están contenidas en la SPA, realizando una única carga inicial y potencialmente solo posponiendo recursos pesados: Grandes cantidades de datos, imágenes, videos, ...

Por tanto con las SPAs eliminamos por completo uno de los principales cuellos de botella: El problema de la latencia (ver artículo de Grigorik). Así podemos conseguir que la aplicación web alcance la velocidad de cualquier aplicación nativa en lo que se refiere al tiempo de espera al cambiar de vistas. Esto se aprecia notablemente sea cual sea el dispositivo, aunque especialmente cuando se utilizan dispositivos móviles.

A su vez, podemos ahorrar mucho ancho de banda y tiempo de proceso de cálculo (en servidor y cliente) si gestionamos el estado de la aplicación desde el cliente con una SPA: Por ejemplo, la SPA del típico e-commerce al cambiar de vista podría mantener en memoria el estado del carrito de compras y no tener que transferir constantemente esa información en el servidor (por supuesto esa información conviene que esté también en el servidor, pero una cosa no está reñida con la otra).

HTML5 y SPAs

Sin lugar a dudas técnicamente se pueden crear SPAs bajo los estándares de HTML4, por tanto dando cobertura a la totalidad de navegadores utilizados hoy en día que tengan soporte de HTML y Javascript.

Sin embargo, algunas APIs incorporadas en el marco de HTML5 aportan soluciones a ciertos retos bastante específicos de las SPAs:

HTML5 History API: Las URLs clásicas eran un problema para las SPAs hasta cierto punto, aunque eso ya no ocurre gracias a esta API. Esto a su vez simplificaría en buena medida el problema de SEO (a pesar de otros intentos de Google como el AJAX Crawling) ya que con PhantomJS en tu servidor puedes renderizar una URL concreta de tu SPA.

HTML5 Application Cache: Si consigues sacarle partido a esta API puedes conseguir que tu aplicación funcione incluso cuando el dispositivo cliente está sin conexión a Internet. Toda una killer feature de las SPA como bien apunta Jakub Mrozek en su charla para la WebExpo 2013.

Herramientas

Hay muchas herramientas para desarrollar SPA. Por mencionar alguna, AngularJS tiene entre otros muchos objetivos crear aplicaciones SPA y a su vez mantener las ventajas del sistema de URLs.

Como este tipo de aplicaciones son bastante complejas (comparado con el uso previo de Javascript), es mejor realizarlas de forma estructurada a partir de un framework como Angular, siguiendo paradigmas de diseño de software similares a MVC.

HTML

Introducción

Un poco de contexto

Tim Berners-Lee creó el HTML original en 1989 para solucionar las deficiencias de los métodos existentes para acceder a información en Internet. Desde que se concibió, encontrar su camino en Internet era una tarea difícil. El contenido en Internet era tratado como documentos individuales, sin que hubiesen métodos sencillos para navegarlos. En esencia, usted tenía que conocer la dirección del documento que estaba buscando e ingresarla manualmente. Para solucionar este problema, Berners-Lee creó dos tecnologías: Hypertext Transfer Protocol (HTTP) y HTML.

HTTP es un protocolo de servicio utilizado para entregar contenido. El comienzo de un URL en su navegador Web (suponiendo que el navegador muestre el URL completo) muy probablemente comenzará con `http://`. Esta parte del URL le dice al navegador qué tipo de protocolo usar cuando esté haciendo la solicitud al servidor Web. Cuando el servidor recibe una solicitud de documento, es probable que ese documento esté escrito o sea convertido a HTML. El documento HTML es lo que se envía de regreso al navegador que hace la solicitud.

HTML es un lenguaje de scripting que le dice al navegador Web cómo presentar el contenido. En este contenido puede haber enlaces a otros documentos, proporcionando un método fácil de usar para navegar entre documentos en Internet.

La combinación de HTTP y HTML ofrece una navegación rápida y fácil por el contenido en Internet, al permitirle simplemente hacer clic en los enlaces de texto para navegar entre documentos. Después de crear estas dos tecnologías, Berners-Lee continuó y fundó el World Wide Web Consortium (W3C). El W3C fue la fuerza guía detrás de las cuatro primeras versiones de HTML.

La intención original de Internet era servir documentos de texto simples. Los primeros navegadores todos estaban basados en texto (sin ventanas lujosas — sólo texto en una pantalla). Incluso la adición de imágenes era un gran problema cuando se introdujo al principio. Ahora, las personas hacen de todo, desde enviar mensajes de e-mail hasta ver televisión en Internet. Internet se ha convertido en mucho más que un mecanismo para transportar documentos de texto simples. Con los recursos y usos llegaron nuevos retos y problemas que el lenguaje HTML nunca fue diseñado para manejar.

El W3C intentó resolver los problemas del Internet de hoy con el estándar Extensible Hypertext Markup Language (XHTML) 2.0. Sin embargo, este estándar no fue bien recibido y fue abandonado en gran medida. En el 2004, mientras el W3C se estaba enfocando en el estándar XHTML 2.0, un grupo llamado el Web Hypertext Application Technology Working Group (WHATWG) comenzó a trabajar en el estándar HTML5, que tuvo una acogida más cálida que el estándar XHTML 2.0. El W3C abandonó el estándar XHTML 2.0 y está trabajando ahora con WHATWG en el desarrollo del HTML5.1.

¿Qué es HTML ?

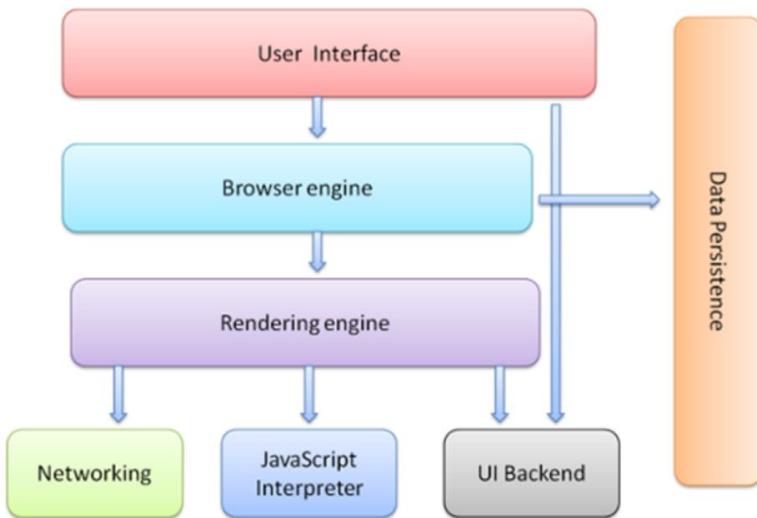
Es el bloque de construcción más básico de la web

- Describe y define el contenido de una página web
- Para describir la apariencia de una página se utilizan hojas de estilo CSS
- Para agregarle funcionalidad se utiliza el lenguaje JavaScript
- HyperText se refiere a vínculos o links que conectan una página a otra
- Utiliza un lenguaje de marcas o “markup” para especificar el texto a mostrar, imágenes, videos y cualquier otro contenido para mostrar en el navegador web.

¿Qué es un servidor web?

- Es un programa que se ejecuta en un equipo servidor que procesa peticiones HTTP generalmente por el puerto 80 y devuelve páginas HTML.
- Posee una estructura de directorios y archivos que constituyen un sitio web
- Los servidores web mas utilizados son:
 - Internet Information Services (IIS8),
 - Apache, Tomcat, NGINX, Node.js
- Lenguajes mas utilizados
 - ASP.NET Web/API, MVC, WebForms
 - PHP
 - Java

Navegadores web - componentes



Interesante presentación del funcionamiento del navegador

https://www.youtube.com/watch?v=A4oNo_nhYA

La función principal de un navegador es solicitar al servidor los recursos web que elija el usuario y mostrarlos en una ventana. El recurso suele ser un documento HTML, pero también puede ser un archivo PDF, una imagen o un objeto de otro tipo. El usuario especifica la ubicación del recurso mediante el uso de una URI (siglas de Uniform Resource Identifier, identificador uniforme de recurso).

La forma en la que el navegador interpreta y muestra los archivos HTML se determina en las especificaciones de CSS y HTML. Estas especificaciones las establece el consorcio W3C (World Wide Web Consortium), que es la organización de estándares de Internet.

Durante años, los navegadores cumplían solo una parte de las especificaciones y desarrollaban sus propias extensiones. Esto provocó graves problemas de compatibilidad para los creadores de contenido web. En la actualidad, la mayoría de los navegadores cumplen las especificaciones en mayor o menor grado.

Fuente: <https://www.html5rocks.com/es/tutorials/internals/howbrowserswork/>

Los componentes principales de un navegador son:

Interfaz de usuario: incluye la barra de direcciones, el botón de avance/retroceso,

el menú de marcadores, etc. (en general, todas las partes visibles del navegador, excepto la ventana principal donde se muestra la página solicitada).

Motor de búsqueda (Browser Engine): coordina las acciones entre la interfaz y el motor de renderización.

Motor de renderización (Rendering Engine): es responsable de mostrar el contenido solicitado. Por ejemplo, si el contenido solicitado es HTML, será el responsable de analizar el código HTML y CSS y de mostrar el contenido analizado en la pantalla.

Red (Networking): es responsable de las llamadas de red, como las solicitudes HTTP. Tiene una interfaz independiente de la plataforma y realiza implementaciones en segundo plano para cada plataforma.

Servidor de la interfaz (UI Backend): permite presentar widgets básicos, como ventanas y cuadros combinados. Muestra una interfaz genérica que no es específica de ninguna plataforma. Utiliza métodos de la interfaz de usuario del sistema operativo en segundo plano.

Intérprete de JavaScript (JavaScript Interpreter): permite analizar y ejecutar el código JavaScript.

Almacenamiento de datos (Data Persistence): es una capa de persistencia. El navegador necesita guardar todo tipo de datos en el disco duro (por ejemplo, las cookies). La nueva especificación de HTML (HTML5) define el concepto de "base de datos web", que consiste en una completa (aunque ligera) base de datos del navegador.

El motor de renderización

La responsabilidad del motor de renderización es "renderizar", es decir, mostrar el contenido solicitado en la pantalla del navegador.

De forma predeterminada, el motor de renderización puede mostrar imágenes y documentos HTML y XML. Puede mostrar otros tipos mediante el uso de complementos (o extensiones); por ejemplo, puede mostrar documentos PDF mediante un complemento capaz de leer archivos PDF.

Motores de renderización

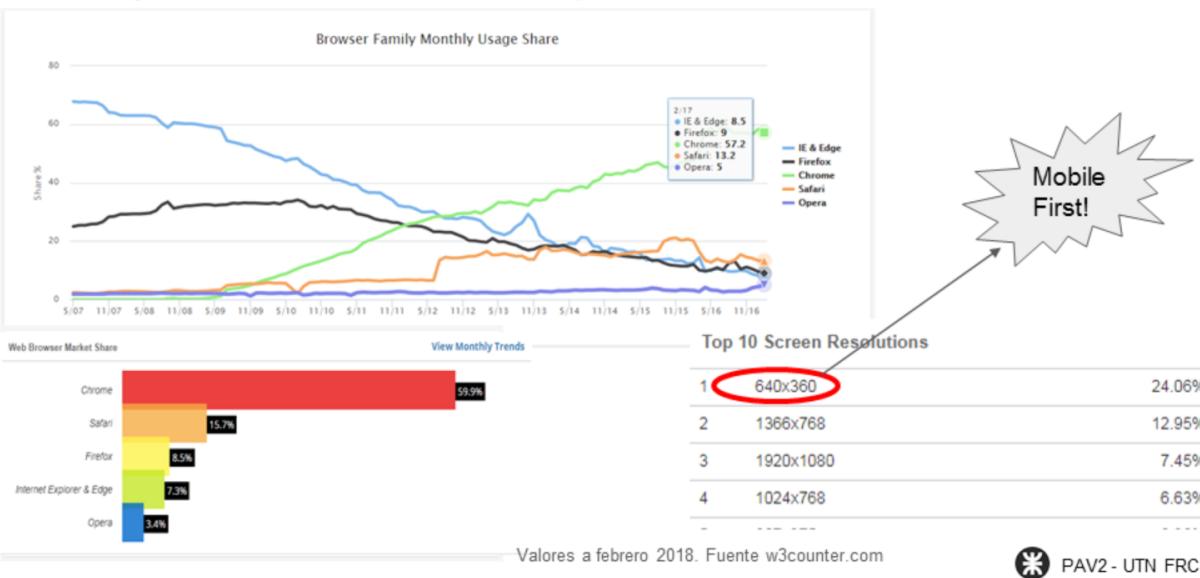
Nuestros navegadores de referencia (Firefox, Chrome y Safari) están basados en dos motores de renderización. Firefox utiliza Gecko, un motor de renderización propio de Mozilla. Tanto Safari como Chrome utilizan WebKit.

Trident es el nombre del [motor de renderizado](#) propietario usado por [Microsoft Internet Explorer](#) 11 para [Windows](#).

EdgeHTML es un motor de renderizado desarrollado por [Microsoft](#) para el navegador [Microsoft Edge](#).

WebKit es un motor de renderización de código abierto que empezó siendo un motor de la plataforma Linux y que fue modificado posteriormente por Apple para hacerlo compatible con Mac y Windows.

Navegadores de internet y resolución de pantalla

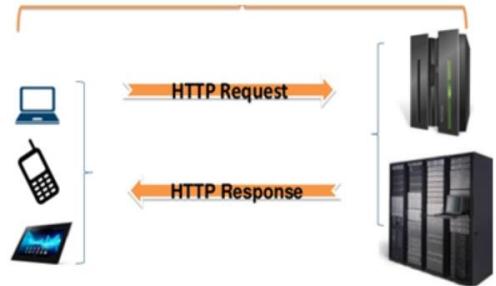


Demo: Cuál es el uso de navegadores en Argentina

- Ingrese a <http://gs.statcounter.com> para ver el mercado de navegadores en argentina y la resolución de pantalla utilizadas

HTTP - Protocolo

- Especifica las reglas de comunicación entre un cliente web y un servidor web
- Protocolo orientado a transacciones
- Sigue el esquema request-response
- El cliente web se denomina user agent
- La petición (request) se realiza enviando un mensaje con un determinado formato. Dicho formato especifica la acción
- La respuesta (response) contiene el documento, archivo, imagen solicitado



Protocolo HTTP (Hypertext Transfer protocol)

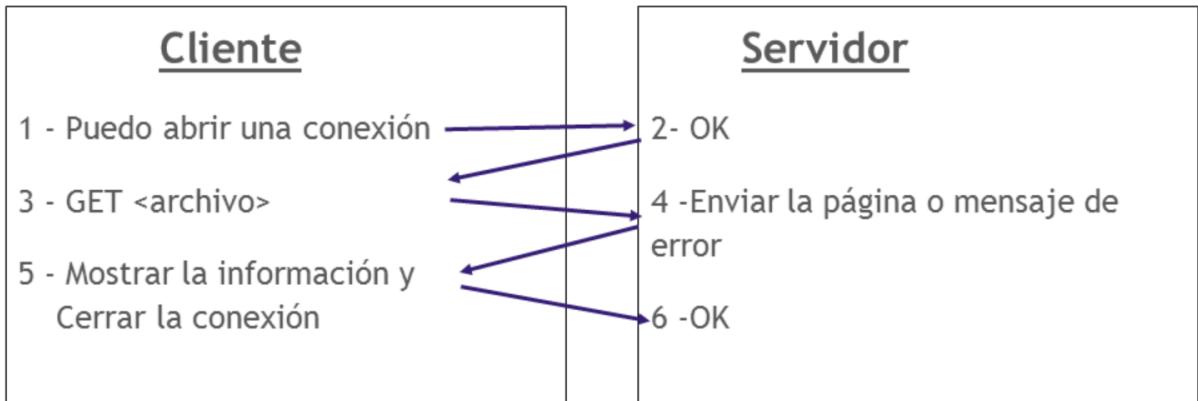
Stateless: cada transacción entre el cliente y el servidor es independiente, es decir, no guarda ninguna información sobre conexiones anteriores.

Para mantener el estado de la aplicaciones web se usan las cookies, que permiten instituir la noción de sesión, y también permite rastrear usuarios

HTTP ha sido diseñado como un **protocolo sin estado** (stateless protocol) lo que significa que cada solicitud (request) o respuesta (Response) es una transacción independiente

Las peticiones al servidor se denominan **request** y las respuestas del servidor **responses**

Una conversación HTTP

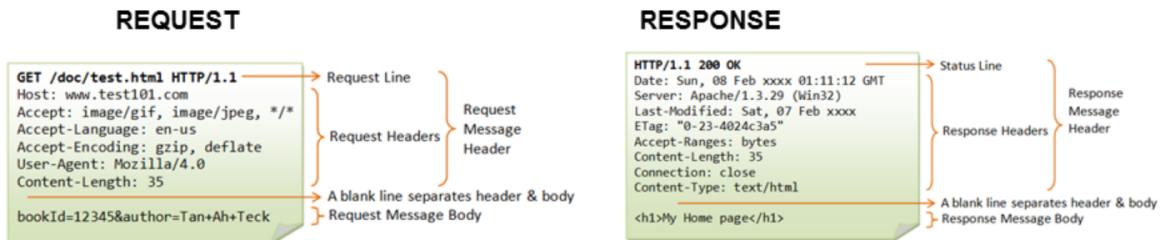


HTTP es el conjunto de reglas que gobiernan el formato y el contenido de una conversación entre un cliente web y un servidor

Mensaje HTTP

Un mensaje del tipo **request** y **response** consta de la siguiente información:

1. Una línea de inicio
2. Opcionalmente una o mas cabeceras del mensaje
3. Opcionalmente el cuerpo del mensaje



Métodos o verbos HTTP

- **GET:** obtiene el recurso especificado por la URL.
- **POST:** envía o somete datos para que sean procesados por el recurso identificado en la URL.
- **PUT:** sube o actualiza un recurso especificado.
- **DELETE:** Borra el recurso especificado.
- **HEAD:** Obtiene sólo las cabeceras de una petición. Es un verbo similar a GET, sólo que GET además de las cabeceras también obtiene el cuerpo de la respuesta.
- **TRACE:** este método se emplea con fines de diagnóstico ya que solicita al servidor que envíe todos los datos de la solicitud enviada.
- **OPTIONS:** sirve para obtener una lista de los métodos HTTP que soporta el servidor
- **CONNECT:** se utiliza para saber si tenemos acceso a un determinado host.

Códigos de estado HTTP

Algunos códigos de estado son:

- 2xx - Successful response
 - 200 OK
 - 201 Created
- 3xx - Redirect response
 - 301 Moved permanently
- 4xx - Client error Response
 - 401 Unauthorized
 - 403 Forbidden
 - 404 File not found
- 5xx - Service error response
 - 500 Server Error
 - 503 Service unavailable

Significados de código en función del número de partida

El primer dígito de cada código de tres dígitos comienza con uno de los números del 1 al 5. Existen los siguientes códigos de estado de 100 a 500:

Los códigos en los 100s son informativos e indican que la solicitud fue recibida y el proceso continúa. Los códigos en los 200s indican la recepción de la solicitud y su procesamiento exitoso. 300s indican la recepción de una solicitud, pero la necesidad de llevar a cabo un paso más para que la solicitud se ha completado. Los 400s simbolizan error del cliente en que la solicitud fue hecha pero la página ha perdido su validez. El 500 indican el error del servidor - hubo una solicitud válida desde el cliente, pero servidor no pudo completar la misma.

Códigos 200 y 301

Código 200 simboliza la solicitud correcta y es la correcta para la mayoría de los escenarios. El código 301 indica que "se trasladó de forma permanente". El recurso se ha dado una nueva dirección URL que es permanente. referencias futuras deben utilizar una de las URL que fueron devueltos. La redirección 301 debe ser utilizado en cualquier momento una URL debe ser redirigido a una URL diferente.

Código 302

El código 302 indica que el servidor está respondiendo a la solicitud con una página ubicación diferente. Sin embargo, el solicitante mantiene mediante la dirección original para futuras peticiones. 302 no es favorable, ya que los

rastreadores de motores de búsqueda tendrá en cuenta la redirección como un arreglo temporal y no proporcionar a la página con el poder de los enlaces de 301 redirecciones.

Códigos 404 y 200

El código 404 indica archivo no encontrado. El servidor no pudo igualar la URL de la solicitud. No hay ninguna indicación del estado de la enfermedad, ya sea permanente o temporal. Esto ocurre cuando el servidor no puede encontrar una solicitud de página que se pongan en venta. Webmasters suelen mostrar texto de error 404 cuando el código de respuesta es de 200. Esto le da a los rastreadores de motores de búsqueda de la indicación de que la página se procesa de forma correcta. A veces, la página web está indexado erróneamente.

Códigos 410 y 404

El código 410 indica que la página no existe y no se dispone de más en el servidor y también que no hay una dirección de reenvío conocido. Esto es más o menos una situación permanente. Los clientes que son capaces de modificar el vínculo necesario suprimir toda referencia a la URL de solicitud tras la aprobación del usuario. Si el servidor no puede determinar el recurso, se debe utilizar el código de estado 404.

código 503

El código de estado 503 indica la incapacidad del servidor de la concesión de la dirección URL solicitada por el mantenimiento o la sobrecarga temporal del servidor. 503 debe ser utilizado en el caso de una interrupción temporal. Se da la indicación de los motores de búsqueda que el sitio es sólo temporal hacia abajo.

Consejos para recordar

Básicamente aquí hay algunos consejos que es absolutamente necesario recordar - 301 redirecciones deben utilizarse en lugar de 302 redirecciones al redirigir las direcciones URL. páginas web que regresan 404 durante largos períodos de tiempo y que tienen vínculos valiosos deben ser redirigidos a otras páginas con el código 301. Cuando los visitantes solicitan páginas que regresan un código de respuesta 404 debe haber personalizado 404 páginas con opciones de navegación.

[Para ver más información](#)

HTML - Versiones

Versión	Año
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.04	1999
XHTML	2000
HTML5	2014

Diferentes versiones de HTML y CSS

Con el tiempo, HTML y CSS han evolucionado. La primera versión de HTML (HTML 1.0) ni siquiera ofrecía la posibilidad de mostrar las imágenes.

Una muy breve historia de estos lenguajes para conocimiento general:

HTML 1: fue la primera versión, creada por Tim Berners-Lee en 1991.

HTML 2: En 1995 se publica el estándar HTML 2.0. A pesar de su nombre, HTML 2.0 es el primer estándar oficial de HTML, es decir, el HTML 1.0 no existió como estándar. HTML 2.0 no soportaba tablas. Se simplificaba al máximo la estructura del documento para agilizar su edición, donde la declaración explícita de los elementos body, html y head es opcional.

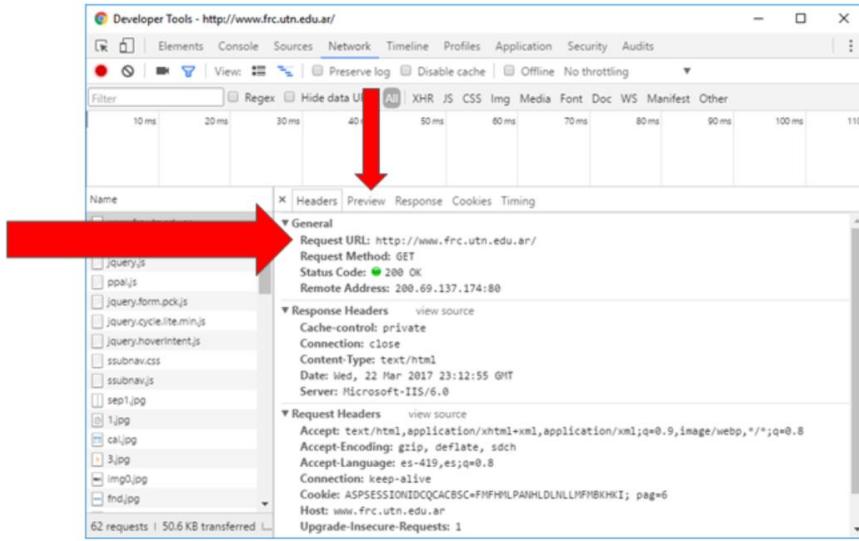
HTML 3.0, esta es la versión que en realidad plantea las bases de las siguientes versiones de HTML. Las reglas y el funcionamiento de esta versión están dadas por el W3C. HTML 3: apareció en 1996, esta nueva versión de HTML, añade muchas posibilidades al lenguaje como tablas, applets, scripts, posicionamiento de texto alrededor de imágenes, etc.

HTML 4: Esta es la versión más común de HTML (en concreto, es HTML 4.01). Apareció por primera vez en 1998 y propone el uso de marcos (que dividen una

página web en varias partes), tablas más complejas, mejoras en las formas, etc. Más importante aún, esta versión permite por primera vez utilizar hojas de estilo del famoso CSS. La última especificación oficial de HTML se publicó en diciembre de 1999 y se denomina HTML 4.01. Desde la publicación de HTML 4.01, el W3C se centró en el desarrollo del estándar XHTML. Por este motivo, en el año 2004, las empresas Apple, Mozilla y Opera mostraron su preocupación por la falta de interés del W3C en HTML y decidieron organizarse en una nueva asociación llamada WHATWG (Web Hypertext Application Technology Working Group) que comenzó el desarrollo del HTML 5, cuyo primer borrador oficial se publicó en enero de 2008. Debido a la fuerza de las empresas que forman el grupo WHATWG y a la publicación de los borradores de HTML 5.0, en marzo de 2007 el W3C decidió retomar la actividad estandarizadora de HTML, dentro del cual decidió integrar el XHTML.

HTML 5: El consorcio internacional W3C, después de una evolución de varios años, liberó el HTML5 como estándar oficial a finales de octubre de 2014. HTML5 incorpora nuevos elementos no contemplados en HTML 4.01. Hay diversos cambios respecto a HTML 4.01. Hay nuevas etiquetas, se introduce la posibilidad de introducir audio y video de forma directa en la web sin necesidad de plugins o complementos en los navegadores, entre otras novedades

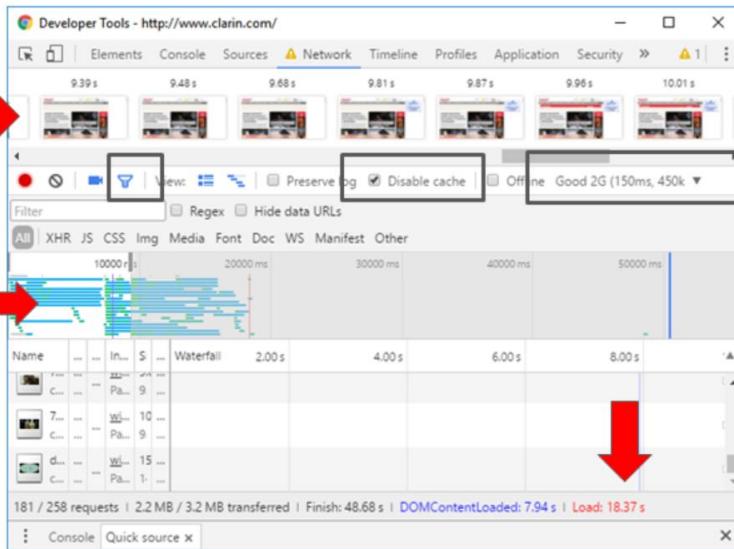
Demo: Chrome DevTools, encabezados, archivos...



Para ver las características de chrome para desarrollo ver el siguiente link:

<https://developers.google.com/web/tools/chrome-devtools/>

Demo: Emular experiencia desde un smartphone



Para mas información ir a <https://developers.google.com/web/tools/chrome-devtools/network-performance/>

Editores de texto para HTML

Editores livianos

1. Bloc de notas
2. Notepad ++
3. Sublime Text
4. Visual Studio Code Editor
5. Atom

Entornos de desarrollo y programación

1. Visual Studio 2017 / 2015 / 2013 update 4
2. Eclipse
3. Netbeans

Nuevas características de HTML 5

Semántica: Permite describir con mayor precisión cuál es su contenido y definir elementos propios de html como así también atributos

Conectividad: Permite comunicarse con el servidor de formas innovadoras.

Sin conexión y almacenamiento: Permite a las páginas web almacenar datos localmente en el lado del cliente y operar sin conexión de manera más eficiente.

Multimedia: soporte contenido multimedia de audio y video nativamente.

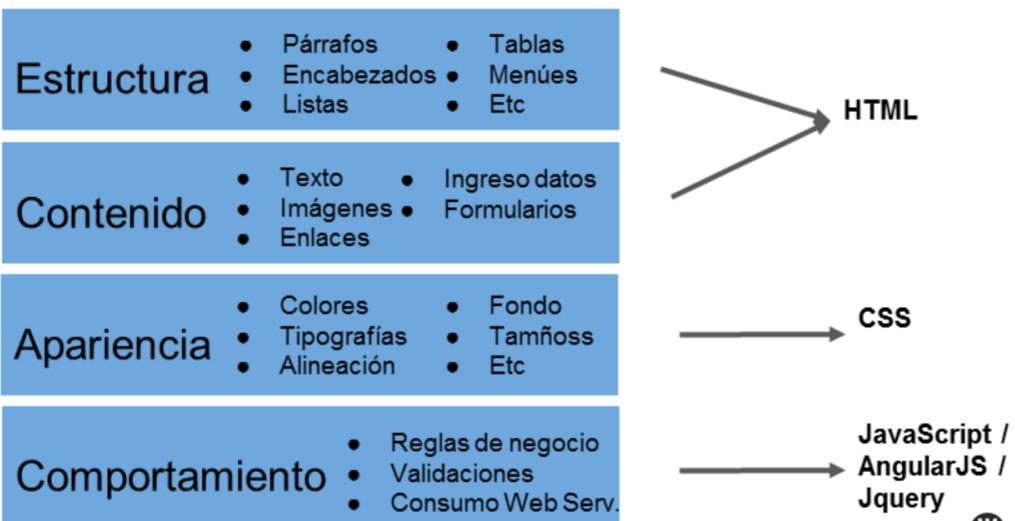
Gráficos y efectos 2D/3D: Incorpora nuevas características que se ocupan de los gráficos en la web como lo son canvas 2D, WebGL, SVG, etc.

Rendimiento e Integración: Proporciona una mayor optimización de la velocidad y un mejor uso del hardware.

Acceso al dispositivo: Proporciona APIs para el uso de varios componentes internos de entrada y salida de nuestro dispositivo, tales como geolocalización

CSS3: Gran variedad de opciones para hacer diseños más sofisticados.

Elementos de un documento HTML



Estructura de un documento HTML5

- La declaración <!DOCTYPE html> define el tipo de documento.
- El texto entre <html> y </html> describe el documento HTML.
- El texto entre <head></head> proporciona información sobre el documento.
- El texto entre <body> y </body> describe el contenido de la página visible en el navegador

```
<!DOCTYPE html>
<html>
  <head>
    <title>Aprendiendo HTML</title>
  </head>
  <body>
    <h1>Bienvenido a PAVII</h1>
    <p>Este es un documento html 5</p>
  </body>
</html>
```

Para qué sirve el tag <head> y <body>

- Head:
 - Es parte fundamental de la estructura de un documento html
 - Se usa siempre
 - No se puede poner dentro de la etiqueta <body>
 - Se incluye información acerca del documento
 - Sección meramente técnica e informativa
 - Dentro de esta sección se pueden establecer
 - Título de la página
 - Configuraciones de Meta Tags
 - Estilos CSS
 - Scripts (javascript y/o enlaces a archivos)
- Body
 - Representa el contenido de un documento HTML
 - Sólo puede haber uno por documento

HEAD traducido del inglés al español significa "cabeza", por eso a esta parte del documento la llamamos cabecera. En el documento HTML comienza con la etiqueta <head> y se indica su final con la etiqueta </head>, se escribe así:

```
<html>
  <head>
    Todo lo que esté aquí pertenece al HEAD
  </head>
  <body>
    </body>
</html>
```

El HEAD es la parte donde se incluye la información acerca del documento, podríamos atrevernos a decir que el HEAD es una sección de un documento HTML meramente "técnica e informativa", pues la mayoría de esta información no la muestra el navegador al usuario e inclusive pudiéramos dejarla vacía y esto no afectaría al funcionamiento o la forma en que se visualiza la página, y si bien el HEAD de un documento HTML pudiera ir vacío siempre es mejor darles la suficiente importancia a las etiquetas que el HEAD contiene, mucho más aun si nuestro objetivo es publicar nuestro trabajo en la web, pues muchas de las etiquetas del HEAD son importantes para los buscadores y para un buen posicionamiento en los resultados de búsqueda.

Bueno vamos al grano y conozcamos las etiquetas del HEAD.

TITLE:

```
<head>
  <title>Título de la página</title>
</head>
```

Un elemento del HEAD visible desde el navegador, muy importante para los buscadores pues es el texto que se visualiza en los resultados de búsqueda. Me parece importante señalar que este elemento nada tiene que ver con el nombre del archivo pues son dos cosas totalmente distintas e independientes una de la otra, por ejemplo podríamos tener un archivo llamado index.html y en el código un <title>Página principal - Bienvenido</title> y esto estaría bien.

META:

```
<head>
  <meta name="description" content="Artículos sobre HTML" />
  <meta name="keywords" content="HTML,manual de HTML" />
  <meta name="author" content="Israel Romero" />
  <meta http-equiv="refresh" content="30" />
</head>
```

Una etiqueta con atributos. La función de esta etiqueta va a depender totalmente de los atributos y valores que contenga, en nuestro ejemplo las primeras 3 tienen los atributos "name" y "content" pero tienen valores diferentes, la primera (con valor "description") indica la descripción de la página, la segunda indica las palabras clave con la que los buscadores deberían de relacionarla, en el ejemplo se pretende que los buscadores muestren la pagina en sus resultados cuando alguien busque "HTML" o "manual de HTML", la tercera incluye el nombre del autor de la página. Ahora la cuarta etiqueta tiene atributos diferentes, en este caso esta etiqueta le dice al navegador que la página se debe actualizar cada 30 segundos. Hay algunos otros argumentos posibles para la etiqueta <meta>, pero el tema principal aquí es el HEAD por lo que tocará hablar del META mas a fondo en otro post.

STYLE:

```
<head>
  <style type="text/css">
    p {color:blue;}
  </style>
</head>
```

Con esta etiqueta se puede incluir código CSS dentro del documento HTML, en este ejemplo se le da un color azul a los párrafos, es decir a los fragmentos de texto contenidos por las etiquetas <p> </p>.

SCRIPT:

```
<head>
  <script type="text/javascript">
    document.write("Hola")
```

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

Otra mas para agregar código de lenguajes ajenos a HTML, en este ejemplo agregamos código de javascript que lo único que hace es mostrarnos un texto que dice "Hola".

Ejemplo tag <head> y <body>

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>PAVII - UTN FRC</title>
    <link href="css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <h1>Ejemplo de tag HEAD para utilizar Bootstrap como hoja de estilo</h1>
    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="js/bootstrap.min.js"></script>
</body>
</html>
```

<meta charset="utf-8"> : Especifica el juego de caracteres (codificación) de la página html

Valores comunes:

UTF-8 - Codificación de caracteres para Unicode

ISO-8859-1 - Codificación de caracteres para el alfabeto latino

En teoría, cualquier codificación de caracteres se puede utilizar, pero no todos los navegadores entienden todas las codificaciones.

para ver mas información ir aqui <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

<meta http-equiv="X-UA-Compatible" content="IE=edge">

El atributo content especifica el modo de la página; por ejemplo, para imitar el comportamiento de Windows Internet Explorer 7, especifique IE=EmulateIE7. Del mismo modo, especifique IE=5, IE=7 o IE=8 para seleccionar uno de estos modos de compatibilidad. También puede especificar IE=edge para indicar a Windows Internet Explorer 8 o 10 que use el máximo modo disponible.

El encabezado X-UA-compatible no distingue entre mayúsculas y minúsculas; no obstante, debe aparecer en el encabezado de la página web (la sección HEAD) antes que todos los demás elementos, excepto el elemento TITLE y otros elementos META.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

La etiqueta meta para el viewport fue introducida por Apple en Safari para móviles, originalmente se utilizó para ayudar a los desarrolladores a mejorar la presentación de sus aplicaciones web en un iPhone, iPod Touch o iPad.

La etiqueta viewport nos permite a los que construimos sitios web o web apps, definir el ancho, alto y escala del área usada por el navegador para mostrar contenido.

Al fijar el ancho o alto del viewport, los desarrolladores podemos usar un número fijo de pixeles (ej: 320, 480, etc) o usar dos constantes, device-width y device height respectivamente. Se considera una buena práctica configurar el viewport con algunas de estas dos constantes, en vez de un ancho o alto fijo.

```
<link href="css/bootstrap.min.css" rel="stylesheet">
```

Esta etiqueta permite enlazar un archivo JavaScript u Hoja de estilo (css) externos con el documento actual. El atributo rel="stylesheet" especifica que el archivo **css/bootstrap.min.css** es de hoja de estilo, y está ubicado en el directorio css del árbol de directorios del sitio web

```
<script
```

```
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
```

Script javascript que está en otro sitio web. En html5 ya no se usa el atributo type="text/javascript". En este caso se posee una referencia a la librería de JQuery que necesita bootstrap.

Observar que el tag **<script>** puede estar en el **<head>** o a finalizar el **<body>** o en ambos lugares.

La colocación de secuencias de comandos en la parte inferior del elemento **<body>** mejora la velocidad de visualización, porque la compilación de secuencias de comandos disminuye la visualización.

Ejemplo

Utilizar el block de notas y desarrollar un documento HTML “Hola Mundo!”

Content delivery network (CDN)



Servidores con copias de datos en varios puntos de la red muy utilizado para librerías javascript

- Reduce la carga de los servidores.
- Red de tráfico distribuida.
- Reduce la latencia.
- Incrementa el ancho de banda.
- Aumenta el web caching.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
```

fuente: wikipedia

Introducción

Las CDN emergen como la solución al actual problema que presenta una web centralizada: lograr bajo tiempo de respuesta y mínima pérdida de información moviendo el contenido de la información más cerca de los usuarios. El objetivo es lograr un equilibrio entre los costos en que incurren los proveedores de contenido web y la calidad de servicio para los usuarios finales.²

Las ventajas de la implementación de este modelo son las siguientes:

Reduce la carga de los servidores.
Red de tráfico distribuida.
Reduce la latencia.
Incrementa el ancho de banda.
Aumenta el web caching.

Arquitectura de una CDN

Componente de entrega de contenidos

Se cuenta con un servidor de origen y un conjunto de servidores sustitutos para replicar el contenido.

Componente de enrutamiento de solicitudes

Usuarios solicitan directamente a los servidores sustitutos.

Interactúa con el componente de distribución para mantener y actualizar el contenido

Componente de distribución de contenido

Mueve el contenido desde el origen a los servidores sustitutos y asegura consistencia

Componente de contabilidad

Mantiene registros de los accesos de los clientes y los registros de uso de los servidores.

Ayuda a la presentación de informes de tráfico y facturación basada en el uso

Servicios y contenidos compatibles con CDN

Contenido estático: Páginas estáticas HTML, imágenes, documentos, parches de software.

Distribución de audio y video por internet: Audio y video en tiempo real. Videos generados por el usuario

Servicio de contenido: Directorio, e – commerce, servicio de transferencia de archivos.

Fuentes de contenido: Grandes empresas, proveedores de servicios Web, compañías de medios de comunicación, y emisoras de noticias

Clientes: Medios de comunicación y empresas de publicidad por internet, centros de datos, proveedores de Internet, minoristas de música en línea, operadores móviles y fabricantes de electrónica de consumo.

Interacción del usuario: Celular, smartphone/PDA, notebooks y computadores de escritorio.

Objetivos de negocio

Escalabilidad

Habilidad para expandirse con el objetivo de manejar nuevos y grandes cantidades de datos. Usuarios y transacciones.

Requiere capacidad para la entrega de contenido dinámico de aprovisionamiento y de alta calidad, con bajo costo operacional

Tendencia futura: los proveedores de contenidos, así como usuarios finales pagarán para obtener contenido de alta calidad

Seguridad

Protección del contenido contra modificaciones y accesos no autorizados.

Requiere red física, software, datos y procedimientos de seguridad.

Tendencia futura: reducir la interrupción del negocio mediante la lucha contra los ataques de negación de servicio y otras actividades maliciosas.

Fiabilidad, Capacidad de respuesta y rendimiento

Disponibilidad de servicios, manejo de posibles interrupciones y experiencia del usuario final.

Requiere una red tolerante a fallas con balanceo de carga adecuada.
Tendencia futura: ubicación del contenido distribuido, la coherencia de caché y los mecanismos de enrutamiento

Beneficios del CDN

Debido a la arquitectura del CDN, se pueden detallar los siguientes beneficios:

Mayor capacidad de conexión.

Disminución del tiempo de respuesta de entrega de información al usuario.

Disminución de los costos asociados a la entrega de contenidos.

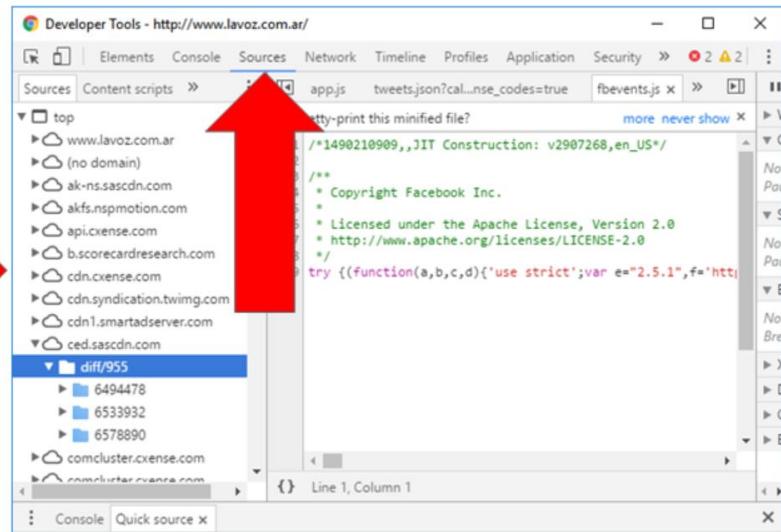
Reducción de la pérdida y demora de paquetes ya que trabajan con nodos cercanos al usuario.

Disminución de carga de la red.

Se tiene 100% de disponibilidad de información, incluso ante la caída de uno de los servidores.

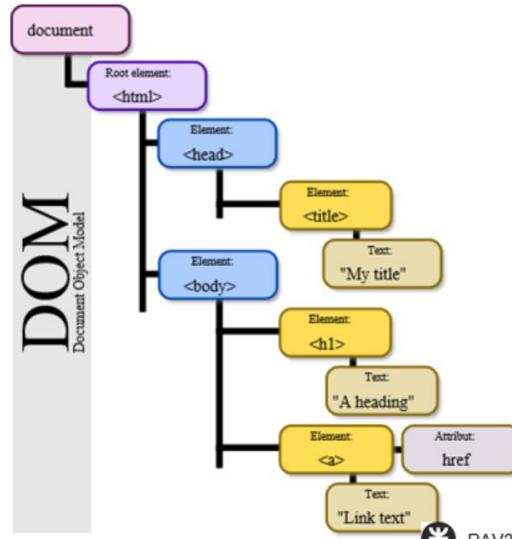
Permite obtener estadísticas de comportamiento de usuarios basado en el registro de páginas visitadas, ubicación geográfica, entre otras.

Demo: mostrar CDN del sitio www.lavoz.com.ar



Qué es el DOM de un documento HTML

- Cuando se carga una página web, el navegador crea un DOM (Document Object Model) de la página.
- El HTML DOM está construido como un árbol de objetos.
- El DOM permite ser manipulado desde JavaScript.



¿Qué es el DOM de HTML?

El DOM HTML es un estándar **de objetos** de modelo y de **programación de interfaces** para HTML. Se define:

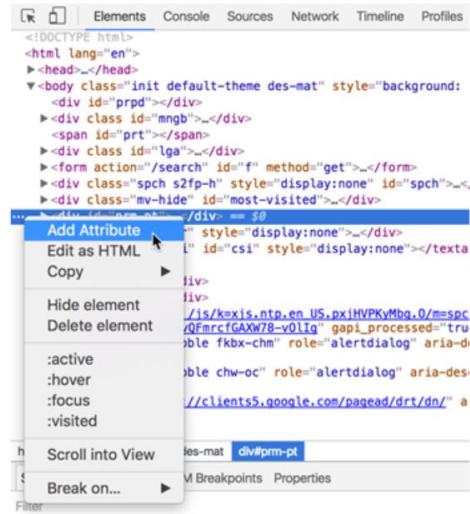
- Los elementos HTML como **objetos**
- Las **propiedades** de todos los elementos HTML
- Los **métodos** para acceder a todos los elementos HTML
- Los **eventos** de todos los elementos HTML

En otras palabras: **El DOM HTML es un estándar de cómo obtener, cambiar, añadir o eliminar elementos HTML.**

Con el modelo de objetos, JavaScript posee toda la funcionalidad necesaria para crear HTML dinámico:

- Puede alterar todos los elementos HTML en la página
- Puede modificar todos los atributos de HTML en la página
- Permite cambiar todos los estilos CSS en la página
- Es posible eliminar elementos y atributos HTML existente
- Contiene la funcionalidad para agregar nuevos elementos y atributos HTML
- Permite implementar los eventos de HTML existentes en la página
- Se puede agregar nuevos manejadores de eventos en la página HTML

Demo: Editar DOM con Chrome DevTools



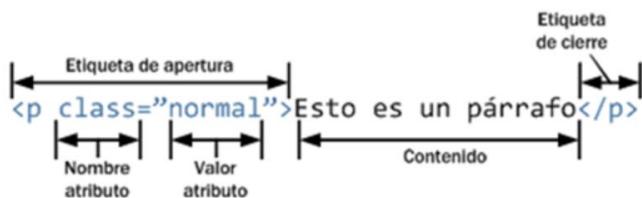
Abrir la página web de la utn y presionar F12 para ver las DevTools de Chrome.
Mostrar el DOM como HTML

Ejercitación

Resolver el ejercicio:

Ejercicio HTML1 - análisis de un documento.docx

Definición de elemento de un documento HTML



- Los elementos son los componentes fundamentales del HTML
- Cuentan con 2 propiedades básicas:
 - Atributos
 - Contenido
- En general se conforman con una Etiqueta de Apertura y otra Cierre.
- Los atributos se colocan dentro la etiqueta de apertura, y el contenido se coloca entre la etiqueta de apertura y la de cierre.

Todos los elementos del estandar HTML5 están listados aquí, descritos por su etiqueta de apertura y agrupados por su función.

Esta lista solamente los elementos válidos de HTML5. Solamente aquellos elementos listados aquí son los que deberían ser usados en nuevos sitios Web.

El simbolo This element was added as part of HTML5 indica que el elemento fue añadido en HTML5. Nótese que otros elementos listados aqui pueden haber sido modificados o extendido en su significado por la especificación HTML5.

Elemento raíz

Elemento	Descripción
<!doctype html>	Define que el documento esta bajo el estandar de HTML 5
Elemento	Descripción
<html>	Representa la raíz de un documento HTML o XHTML. Todos los demás elementos deben ser descendientes de este elemento.

Metadatos del documento

Elemento	Descripción
<head>	Representa una colección de metadatos acerca del documento, incluyendo enlaces a, o definiciones de, scripts y hojas de estilo.
<title>	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.

<base> Define la URL base para las URLs relativas en la página.
<link> Usada para enlazar JavaScript y CSS externos con el documento HTML actual.
<meta> Define los metadatos que no pueden ser definidos usando otro elemento HTML.
<style> Etiqueta de estilo usada para escribir CSS en línea.

Scripting

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

Secciones

Elemento	Descripción
<body>	Representa el contenido principal de un documento HTML. Solo hay un elemento <body> en un documento.
<section>	(HTML5)Define una sección en un documento.
<nav>	(HTML5)Define una sección que solamente contiene enlaces de navegación
<article>	(HTML5)Define contenido autónomo que podría existir independientemente del resto del contenido.
<aside>	(HTML5)Define algunos contenidos vagamente relacionados con el resto del contenido de la página. Si es removido, el contenido restante seguirá teniendo sentido
<h1>,<h2>, <h3>,<h4> , <h5>,<h6>	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.

<header>	(HTML5)Define la cabecera de una página o sección. Usualmente contiene un logotipo, el título del sitio Web y una tabla de navegación de contenidos.
<footer>	(HTML5)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.
<address>	Define una sección que contiene información de contacto.
<main>	(HTML5)Define el contenido principal o importante en el documento. Solamente existe un elemento <main> en el documento.

Agrupación de Contenido

Elemento	Descripción
<p>	Define una parte que debe mostrarse como un párrafo.

<hr> Representa un quiebre temático entre párrafos de una sección o artículo o cualquier contenido.

<pre> Indica que su contenido está preformatado y que este formato debe ser preservado.

<blockquote> Representa una contenido citado desde otra fuente.

 Define una lista ordenada de artículos.

 Define una lista de artículos sin orden.

 Define un artículo de una lista ennumerada.

<dl> Define una lista de definiciones, es decir, una lista de términos y sus definiciones asociadas.

<dt> Representa un término definido por el siguiente <dd> .

<dd> Representa la definición de los términos listados antes que él.

<figure> (HTML5)Representa una figura ilustrada como parte del documento.

<figcaption> (HTML5)Representa la leyenda de una figura.

<div> Representa un contenedor genérico sin ningún significado especial.

Semántica a nivel de Texto

Elemento Descripción

<a> Representa un hiperenlace , enlazando a otro recurso.

 Representa un texto enfatizado , como un acento de intensidad.

 Representa un texto especialmente importante .

<small> Representa un comentario aparte , es decir, textos como un descargo de responsabilidad o una nota de derechos de autoría, que no son esenciales para la comprensión del documento.

<s> Representa contenido que ya no es exacto o relevante .

<cite> Representa el título de una obra .

<q> Representa una cita textual inline.

<dfn> Representa un término cuya definición está contenida en su contenido ancestro más próximo.

<abbr> Representa una abreviación o un acrónimo ; la expansión de la abreviatura puede ser representada por el atributo title.

<data> (HTML5)Asocia un equivalente legible por máquina a sus contenidos. (Este elemento está sólamente en la versión de la WHATWG del estandar HTML, y no en la versión de la W3C de HTML5).

<time> (HTML5)Representa un valor de fecha y hora; el equivalente legible por máquina puede ser representado en el atributo datetime.

<code> Representa un código de ordenador .

<var> Representa a una variable, es decir, una expresión matemática o contexto de programación, un identificador que represente a una constante, un símbolo que identifica una cantidad física, un parámetro de una función o un marcador de posición en prosa.

<samp> Representa la salida de un programa o un ordenador.

<kbd> Representa la entrada de usuario, por lo general desde un teclado, pero no necesariamente, este puede representar otras formas de entrada de usuario,

como comandos de voz transcritos.

<sub>,**<sup>** Representan un subíndice y un superíndice, respectivamente.

<i> Representa un texto en una voz o estado de ánimo alterno, 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.

**** Representa un texto hacia el cual se llama la atención para propósitos utilitarios. No confiere ninguna importancia adicional y no implica una voz alterna.

<u> Representa una anotación no textual sin-articular, como etiquetar un texto como mal escrito o etiquetar un nombre propio en texto en Chino.

<mark> (HTML5)Representa texto resaltado con propósitos de referencia, es decir por su relevancia en otro contexto.

<ruby> (HTML5)

Representa contenidos a ser marcados con anotaciones ruby, 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 furigana Japonés.

<rt> (HTML5)Representa el texto de una anotación ruby .

<rp> (HTML5)Representa los paréntesis alrededor de una anotación ruby, usada para mostrar la anotación de manera alterna por los navegadores que no soporten despliegue estandar para las anotaciones.

<bdi> (HTML5)Representa un texto que debe ser aislado de sus alrededores para el formateado bidireccional del texto. Permite incrustar un fragmento de texto con una direccionalidad diferente o desconocida.

<bdo> Representa la direccionalidad de sus descendientes con el fin de anular de forma explícita al algoritmo bidireccional Unicode.

**** Representa texto sin un significado específico. Este debe ser usado cuando ningún otro elemento semántico le confiere un significado adecuado, en cuyo caso, provendrá de atributos globales como class, lang, o dir.

**
** Representa un salto de línea.

<wbr> (HTML5)Representa una oportunidad de salto de línea, es decir, un punto sugerido de envoltura donde el texto de múltiples líneas puede ser dividido para mejorar su legibilidad.

Ediciones

Elemento Descripción

<ins> Define una adición en el documento.

**** Define una remoción del documento.

Contenido incrustado

Elemento Descripción

**** Representa una imagen.

<iframe> Representa un contexto anidado de navegación, es decir, un documento HTML embebido.

<embed> (HTML5) Representa un punto de integración para una aplicación o contenido interactivo externo que por lo general no es HTML.

<object> Representa un recurso externo, que será tratado como una imagen, un sub-documento HTML o un recurso externo a ser procesado por un plugin.

<param> Define parámetros para el uso por los plugins invocados por los elementos **<object>**.

<video> (HTML5) Representa un video , y sus archivos de audio y capciones asociadas, con la interfaz necesaria para reproducirlos.

<audio> (HTML5) Representa un sonido o stream de audio.

<source> (HTML5) Permite a autores especificar recursos multimedia alternativos para los elementos multimedia como **<video>** o **<audio>**.

<track> (HTML5) Permite a autores especificar una pista de texto temporizado para elementos multimedia como **<video>** o **<audio>**.

<canvas> (HTML5) Representa un área de mapa de bits en el que se pueden utilizar scripts para renderizar gráficos como gráficas, gráficas de juegos o cualquier imagen visual al vuelo.

<map> En conjunto con **<area>**, define un mapa de imagen.

<area> En conjunto con **<map>**, define un mapa de imagen.

<svg> (HTML5) Define una imagen vectorial embebida.

<math> (HTML5) Define una fórmula matemática.

Datos tabulares

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

Formularios

| Elemento | Descripción |
|-------------------------|---|
| <form> | Representa un formulario, consistiendo de controles que puede ser enviado a un servidor para procesamiento. |
| <fieldset> | Representa un conjunto de controles. |
| <legend> | Representa el título de un <fieldset> . |
| <label> | Representa el título de un control de formulario. |
| <input> | Representa un campo de datos escrito que permite al usuario editar |

los datos.

| | |
|------------|--|
| <button> | Representa un botón . |
| <select> | Representa un control que permite la selección entre un conjunto de opciones. |
| <datalist> | (HTML5)Representa un conjunto de opciones predefinidas para otros controles. |
| <optgroup> | Representa un conjunto de opciones, agrupadas lógicamente. |
| <option> | Representa una opción en un elemento <select> , o una sugerencia de un elemento <datalist> . |
| <textarea> | Representa un control de edición de texto multi-línea. |
| <keygen> | (HTML5)Representa un control de par generador de llaves. |
| <output> | (HTML5)Representa el resultado de un cálculo. |
| <progress> | (HTML5)Representa el progreso de finalización de una tarea. |
| <meter> | (HTML5)Representa la medida escalar (o el valor fraccionario) dentro de un rango conocido. |

Elementos interactivos

Elemento Descripción

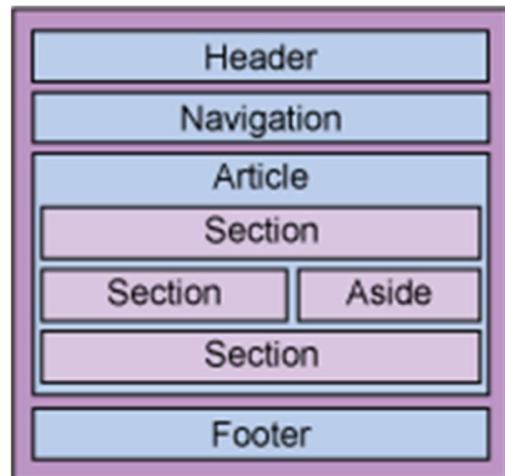
| | |
|-----------|--|
| <details> | (HTML5)Representa un widget desde el que un usuario puede obtener información o controles adicionales. |
| <summary> | (HTML5)Representa un resumen, título o leyenda para un elemento <details> dado. |
| <command> | (HTML5)Representa un comando que un usuario puede invocar. |
| <menu> | (HTML5)Representa una lista de comandos . |

Recomendaciones para escritura de html

- Uso minúsculas en nombres de elementos
- HTML5 permite mezclar letras mayúsculas y minúsculas en nombres de elementos. Se recomienda utilizar nombres de elementos minúsculas porque: mezclar mayúsculas y minúsculas los nombres es una mala práctica en HTML
- El texto en minúsculas es más limpio y fácil de leer
- Encerrar los valores de los atributos siempre con comilla doble
- Evitar espacios entre el signo igual (=) en la especificación del atributo="valor" en los elementos de HTML.

Elementos estructurales de HTML

- header
- section
- article
- aside
- footer
- nav



<https://www.ibm.com/developerworks/ssa/web/library/wa-html5structuraltags/>

Nuevos elementos estructurales

La razón para crear nuevas etiquetas estructurales es dividir las páginas Web en partes lógicas que describan el tipo de contenido que incluyen. Conceptualmente, piense en la página Web como un documento. Los documentos tienen encabezados, pies de página, capítulos y otras convenciones diferentes que dividen el documento en partes lógicas.

Esta sección revisa los métodos actuales de dividir un documento HTML usando código genérico de muestra. Durante el resto de este artículo, usted revisará el código original usando las nuevas etiquetas estructurales HTML5 para ver paso a paso cómo el documento es transformado en secciones lógicas.

Enfoque HTML 4

Si usted ha creado incluso los documentos HTML más simples, entonces estará familiarizado (a) con la etiqueta div . La etiqueta div es el principal mecanismo de la era pre-HTML5 para crear bloques de contenido en un documento HTML. Por ejemplo, el [Listado 2](#) muestra cómo usted puede usar etiquetas div para crear una página simple con un encabezado, un área de contenido y un pie de página.

Listado 2. Página HTML simple usando etiquetas div

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html>  
<head>  
<title>  
Una página HTML simple usando Divs  
</title>  
</head>  
<body>  
<div id='header'>Header</div>  
<div id='content'>Content</div>  
<div id='footer'>Footer</div>  
</body>  
</html>
```

Esto funciona bien; la etiqueta div es una buena etiqueta de propósito general. Sin embargo, además de mirar el atributo id de cada etiqueta div, es difícil decir qué sección del documento representa cada etiqueta div. Aunque usted puede argumentar que la id es un indicador suficiente si se nombra adecuadamente, los atributos id son arbitrarios. Hay muchas variaciones que se pueden considerar id's igualmente válidas. La etiqueta en sí no ofrece ninguna indicación sobre el tipo de contenido que se pretende que represente.

Enfoque HTML5

HTML5 responde a este problema proporcionando un conjunto de etiquetas que definen con mayor claridad los bloques principales de contenido que componen un documento HTML. Sin importar el contenido final mostrado por la página Web, la mayoría de páginas Web consisten en combinaciones de variantes de secciones y elementos de página comunes.

El código siguiente crea una página simple con un encabezado, un área de contenido y un pie de página. Estos y otras secciones y elementos de página son bastante comunes, de manera que HTML5 incluye etiquetas que dividen los documentos en las secciones comunes y que indican el contenido de cada una. Las nuevas etiquetas son:

[header](#)
[section](#)
[article](#)

[aside](#)

[footer](#)

[Nav](#)

El área header .

Como su nombre lo sugiere, la etiqueta header tiene por objeto marcar una sección de la página HTML como el encabezado. Listado 3 muestra el ejemplo de código del [Listado 2](#) modificado para que use un header .

Listado 3. Añadiendo una etiqueta header .

```
<!DOCTYPE html>
<html>
<head>
<title>
Una página HTML simple
</title>
</head>
<body>
<header>Header</header>
<div id='content'>Content</div>
<div id='footer'>Footer</div>
</body>
</html>
```

El doctype en el Listado 3 también se cambió para indicar que el navegador debería utilizar HTML5 para presentar la página. Desde este punto en adelante, todos los ejemplos suponen que usted está utilizando el doctype correcto.

El área section .

La etiqueta section tiene por objeto identificar porciones significativas del contenido de la página. Esta etiqueta es de alguna forma análoga a dividir un libro en capítulos. Añadiendo una etiqueta section al código de ejemplo da como resultado el código en el Listado 4.

Listado 4. Añadiendo una etiqueta section .

```
<!DOCTYPE html>
<html>
<head>
<title>
```

```
Una página HTML simple
</title>
</head>
<body>
<header>Header</header>
<section>
    <p>
        Esta es una sección importante de la página.
    </p>
</section>
<div id='footer'>Footer</div>
</body>
</html>
```

El área article .

La etiqueta article identifica las secciones principales del contenido dentro de la página Web. Piense en un blog, donde cada publicación de cada individuo constituye una porción significativa de contenido. Añadiendo etiquetas article al código de ejemplo da como resultado el código en el Listado 5.

Listado 5. Añadiendo etiquetas article

```
<!DOCTYPE html>
<html>
<head>
<title>
    Una página HTML simple
</title>
</head>
<body>
<header>Header</header>
<section>
<article>
    <p>
        Esta es una sección importante del contenido de la página.
        Tal vez una publicación en blog.
    </p>
</article>
```

```
<article>
<p>
    Esta es una sección importante del contenido de la página.
    Tal vez una publicación en blog.
</p>
</article>

</section>
<div id='footer'>Footer</div>
</body>
</html>
```

La etiqueta aside.

La etiqueta aside indica que el contenido dentro de ella está relacionado el contenido principal de la página pero que no es parte de ella. En cierta forma es análogo a usar paréntesis para hacer un comentario en un cuerpo de texto (como este). El contenido entre paréntesis proporciona información adicional sobre el elemento que lo contiene. Añadiendo una etiqueta aside al código de ejemplo da como resultado el código en el Listado 6.

Listado 6. Añadiendo una etiqueta aside .

```
<!DOCTYPE html>
<html>
<head>
<title>
    Una página HTML simple
</title>
</head>
<body>
<header>Header</header>
<section>
    <article>
        <p>
            Esta es una sección importante del contenido de la página.
            Tal vez una publicación en blog.
        </p>
    </article>
    <article>
```

```
<p>
Esta es una sección importante del contenido de la página.
Tal vez una publicación en blog.
</p>
</article>

</section>
<div id='footer'>Footer</div>
</body>
</html>
```

La etiqueta footer .

La etiqueta footer marca el contenido dentro del elemento que es el pie de página del documento. Añadiendo una etiqueta footer al código de ejemplo da como resultado el código en el Listado 7.

Listado 7. Añadiendo una etiqueta footer .

```
<!DOCTYPE html>
<html>
<head>
<title>
Una página HTML simple
</title>
</head>
<body>
<header>Header</header>
<section>
<article>
<p>
Esta es una sección importante del contenido de la página.
Tal vez una publicación en blog.
</p>
<aside>
</p>
Este es un aparte de la primera publicación en blog.
</p>
</aside>
</article>
</section>
<div id='footer'>Footer</div>
</body>
</html>
```

```
<article>
  <p>
    Esta es una sección importante del contenido de la página.
    Tal vez una publicación en blog.
  </p>
</article>
</section>
<footer>
  Footer
</footer>
</body>
</html>
en este punto, todas las etiquetas div originales han sido reemplazadas con etiquetas HTML5 estructurales.
```

La etiqueta nav .

El contenido dentro de la etiqueta nav tiene por objeto propósitos de navegación. Añadiendo una etiqueta nav al código de ejemplo da como resultado el código en el Listado 8.

Listado 8. Añadiendo una etiqueta nav .

```
<!DOCTYPE html>
<html>
<head>
<title>
  Una página HTML simple
</title>
</head>
<body>
<header>Header
  <nav>
    <a href='#'>Algún enlace de navegación</a>
    <a href='#'>Algún enlace de navegación adicional</a>
    <a href='#'>Un tercer enlace de navegación</a>
  </nav>
</header>
<section>
```

```
<article>
<p>
Esta es una sección importante del contenido de la página.
Tal vez una publicación en blog.
</p>
</aside>
</p>
Este es un aparte de la primera publicación en blog.
</p>
</aside>
</article>
<article>
<p>
Esta es una sección importante del contenido de la página.
Tal vez una publicación en blog.
</p>
</article>
</section>
<footer>Footer</footer>
</body>
</html>
```

Conclusión

Las nuevas etiquetas HTML5 describen los tipos de contenido que contienen, y ayudan a dividir el documento en secciones lógicas. Todavía depende de usted decidir cuándo y dónde utilizar las nuevas etiquetas dentro de un documento, de forma similar a como un autor escribe un libro. Mientras dos autores escribiendo el mismo libro pueden optar por diferentes formas de dividir el libro en capítulos, la acción de usar capítulos ofrece un método consistente de dividir el libro en secciones. De manera similar, aunque los dos autores de una página Web dada pueden optar por estructuras diferentes, las nuevas etiquetas estructurales HTML5 proporcionan nuevas convenciones que los desarrolladores de páginas Web pueden usar y que las viejas etiquetas div no ofrecían.

Elementos HTML más utilizados

| Elemento | Descripción |
|---|--|
| <code><h1>, <h2>, <h3>, <h4>, <h5>, <h6></code> | Los elementos de cabecera implementan seis niveles de cabeceras de documentos; <code><h1></code> es la de mayor y <code><h6></code> es la de menor importancia. Un elemento de cabecera describe brevemente el tema de la sección que introduce. |
| <code><p></code> | Define una parte que debe mostrarse como un párrafo. |
| <code></code> | Negrita |
| <code>
</code> | Salto de línea |
| <code><label></code> | Representa el título de un control de formulario. |
| <code><div></code> | contenedor |

Caracteres especiales de texto en HTML

En HTML los caracteres propios de los idiomas diferentes al inglés pueden ser problemáticos.

La correcta visualización depende de la codificación del documento UTF-8, del editor de HTML, y del servidor.

Para evitar problemas de visualización con los caracteres reservados de HTML se debe utilizar el carácter de **entidad** HTML.

```
<p>Esto es una frase: "Que los a&ntilde;os no te hagan  
m&aacute;s viejo, sino m&aacute;s sabio"</p>
```

| Entidad HTML | Carácter |
|--------------|----------|
| &nbsp | espacio |
| ñ | ñ |
| Ñ | Ñ |
| á | á |
| é | é |
| í | í |
| ó | ó |
| ú | ú |
| " | " |
| < | < |
| > | > |



HTML Listas

Permiten estructurar mejor el texto y para ordenar la información. Hay dos tipos de lista

- Listas no ordenadas o listas de viñetas

- C#
- Entity Framework
- ASP.NET WebAPI
- AngularJS

```
<h1>Contenido sin ordenar PAVII</h1>
<ul>
<li>C#</li>
<li>Entity Framework</li>
<li>ASP.NET WebAPI</li>
<li>AngularJS</li>
</ul>
```

- Listas ordenadas o numeradas

1. C#
2. Entity Framework
3. ASP.NET WebAPI
4. AngularJS

```
<h1>Contenido ordenado PAVII</h1>
<ol>
<li>C#</li>
<li>Entity Framework</li>
<li>ASP.NET WebAPI</li>
<li>AngularJS</li>
</ol>
```

HTML - Imágenes

Se especifican los siguientes atributos:

- El archivo a visualizar (src),
- texto alternativo, o para personas con discapacidad visual (alt),
- width, y height

```

```



HTML - Enlaces

Define un hiper vínculo a una ubicación en la misma página o cualquier otra página en la Web.

```
<h1>Enlace con texto</h1>
<a href="http://www.frc.utn.edu.ar">Ir al sitio de la UTN - FRC</a>

<h1>Enlace con imagen</h1>
<a href="http://www.institucional.frc.utn.edu.ar/sistemas/">
    
</a>

<h1>Enlace a un teléfono, muy &acute;til en smartphones</h1>
<a href="tel:+54351598-6000">Para llamar a la facultad 598-6000</a>
```

Enlace con texto

[Ir al sitio de la UTN - FRC](http://www.frc.utn.edu.ar)

Enlace con imagen



Enlace a un teléfono, muy

[Para llamar a la facultad 598-6000](tel:+54351598-6000)

Ejercitación

Resolver el ejercicio

Ejercicio HTML2 - formateo básico.docx

HTML -Tablas

```
<table>
  <tr>
    <th>Documento</th>
    <th>Apellido y Nombre</th>
  </tr>
  <tr>
    <td>34.434.532</td>
    <td>Perez Juan</td>
  </tr>
  <tr>
    <td>32.524.922</td>
    <td>Lopez Mar&iacute;a</td>
  </tr>
  <tr>
    <td colspan="2">Total de
    clientes: 2</td>
  </tr>
</table>
```



| Documento Apellido y Nombre |
|-----------------------------|
| 34.434.532 Perez Juan |
| 32.524.922 Lopez María |
| Total de clientes: 2 |

- <table> indica que se debe renderizar una tabla.
- Cada fila es un tag <tr> (**table row**)
- La cabecera se especifica como <th>
- Cada celda es un tag <td> (**table data**)
- El atributo **colspan** indica que una celda va a ocupar “n” columnas. Por defecto colspan=”1”

Ejercitación

Resolver el ejercicio:

Ejercicio HTML3 - tablas.docx

HTML - Elementos de línea y de bloque

- Elementos **inline**: Se posiciona horizontalmente en línea con los otros elementos.
 - La altura y el ancho se define en base al contenido que posea.
 - Solo puede contener elementos de tipo inline.
 - No se puede aplicar una anchura y un altura fija por medio de CSS.
 - Solo ocupan el espacio necesario para mostrar sus contenidos
 - Ej.: [<a>](#),

, ,  <i>, ****, <big>, <small>, <u>, ~~<s>~~, **, ****, <input>, <select>, <textarea>, <label>, <button>
- Elementos **block**: Forma un bloque y se posiciona de forma vertical con un nuevo salto de línea.
 - Las anchura es la máxima que puede tomar dentro de su elemento contenedor (padre)
 - La altura cambia en base al contenido que posea.
 - Puede contener otros elementos de tipo inline y block
 - Por medio de CSS se le puede aplicar una anchura y un altura fija.
 - Ej:

<p>

,

<h1>

,

<h2>

,

<h3>

,

<h4>

,

<h5>

,

<h6>

,

<div>

,
,
, -
,

, <form>

<big> No es soportado en HTML5

Se puede consultar los InLine en https://www.w3schools.com/html/html_blocks.asp

Se puede consultar los tags en <https://www.w3schools.com/tags/>

HTML - Etiqueta <div>

- Define un bloque de contenido o sección de la página.
- Puede contener otros elementos tipo block o inline de html
- Muy utilizado para el maquetado de la página y para aplicar hojas de estilo
- El atributo `style="color:red"` especifica el color

```
<div style="color:red">
    <h3>Esto es una cabecera</h3>
    <p>El párrafo está en rojo.</p>
</div>
<div style="color:blue">
    <h3>Esto es una cabecera</h3>
    <p>El párrafo está en azul.</p>
</div>
```



Esto es una cabecera
El párrafo está en rojo.

Esto es una cabecera
El párrafo está en azul.

Demo: Análisis de una página html

- Para analizar que un documento conforme el estándar HTML ir a
<https://validator.w3.org/>
- Para recomendaciones para mejorar la velocidad de carga de la página ir a
<https://developers.google.com/speed/pagespeed/>

Minificar recursos (HTML, CSS y JavaScript)

- Técnica que permite la eliminación de bytes innecesarios, como los espacios adicionales, saltos de línea y sangrías en HTML, CSS y JavaScript
- Permite acelerar la descarga, el análisis y el tiempo de ejecución
- En CSS y JavaScript, es posible reducir aún más el tamaño del archivo al cambiar el nombre de las variables.
- En JavaScript y CSS los archivos minificados aparecen con la extensión .min.js o .min.css
- Existen varios minificadores online y analizadores como:
 - PageSpeed Insights de google
 - cssmin.js
 - Closure Compiler

Minificar recursos (HTML, CSS y Javascript)

Información general

La minificación de recursos se refiere a la eliminación de bytes innecesarios, como los espacios adicionales, saltos de línea y sangrías. Al minimizar los códigos HTML, CSS y JavaScript, es posible acelerar la descarga, el análisis y el tiempo de ejecución. Además, en CSS y en JavaScript, es posible reducir aun más el tamaño del archivo al cambiar el nombre de las variables, siempre y cuando el código HTML esté actualizado correctamente para garantizar que los selectores sigan funcionando.

Recomendaciones

Debes minificar el código HTML, CSS y JavaScript.

Para minificar HTML, puedes utilizar la [extensión de PageSpeed Insights para Chrome](#) con la que generar una versión optimizada del código HTML. Analiza tu página HTML y busca la regla "Minificar HTML". Haz clic en "Ver el contenido optimizado" para obtener el código HTML optimizado.

Para minificar CSS, usa [YUI Compressor](#) y [cssmin.js](#).

Para minificar JavaScript, prueba [Closure Compiler](#), [JSMin](#) o [YUI Compressor](#). Se puede crear un proceso de construcción que utilice estas herramientas para minificar y cambiar el nombre de los archivos de desarrollo, y guardarlos en un directorio de producción.

HTML - Formularios

Es un elemento de html definido por la etiqueta <form> que permite agrupar múltiples **elementos de ingreso de datos y botones**

Atributos más utilizados

- **method**: indica cómo se envían los valores del formulario. POST y GET
- **action**: indica la ubicación hacia donde será enviada la información

Sólo se enviará la información de los elementos de ingreso de datos

```
<form method="post" action="producto.html">
    <!--aquí van los elementos para ingreso de datos que se envían a la página
    especificada en el atributo action--&gt;
&lt;/form&gt;</pre>
```

Los formularios están delimitados con la etiqueta <form>, que permite reunir varios elementos de un formulario, como botones y casillas de texto.

La etiqueta form tiene dos atributos muy importantes:

method

Indica la manera en que serán enviados los valores del formulario. Los métodos permitidos son POST y GET.

POST Envía los datos de manera oculta. GET envía los datos agregándolos a la dirección URL y los separa de la URL base con un signo de interrogación

action

Indica la ubicación a la cual será enviada la información. Puede ser un correo electrónico un script, inclusive, otra página HTML

```
<form method="post" action="file2.html"></form>
```

Dentro de un formulario se pueden insertar cualquier elemento HTML en como texto, botones, tablas y enlaces, pero los elementos interactivos son los más interesantes

HTML - Elementos de ingreso de datos

FULL COMPATIBLE

- input (previo a html 5)
 - text
 - password
 - button
 - submit
 - reset
 - radio
 - checkbox
- textarea
- button
- select

PARCIALMENTE COMPATIBLE

- input
 - color
 - date, datetime, datetime-local, month
 - email
 - number
 - range
 - search
 - tel
 - time
 - url
 - week

<http://caniuse.com/#search=input>

Atención: no todos los input de html5 son compatibles con los últimos navegadores.

En este link <http://caniuse.com/#search=input> se puede verificar la compatibilidad con los diferentes navegadores y versiones de los elementos de html.

HTML - atributos nuevos para <input>

HTML 4

value
readonly
disabled
size
maxlength

HTML 5

autocomplete
autofocus
form
formaction
formenctype



HTML 5

formmethod
formnovalidate
formtarget
height and width
list
min and max
multiple
pattern (regexp)
placeholder
required
step



PAV2 - UTN FRC

En la siguiente pagina se puede ver información de los atributos:

https://www.w3schools.com/html/html_form_attributes.asp

HTML - Elemento <input> parte 1

Permite ingreso de datos en un cuadro de texto

```
<input type="text" name="Nombre" required maxlength="45" />
```

Para el ingreso de contraseña:

```
<input type="password" name="Clave" placeholder="Clave" value="" />
```

Para establecer valores no visibles al usuario pero que se envían al servidor

```
<input type="hidden" value="valor oculto" name="id" />
```

Permite definir un botón que es una imagen

```
<input type="image" src="img_submit.gif" alt="Submit">
```

Atributos:

type:

- text: texto claro
- password: El ingreso de los caracteres aparece en la interfaz del navegador de manera oculta, pero a través del DOM se puede acceder al valor que contiene

name: identificador para trabajar con el DOM

required: requerido

maxlength: cantidad máxima de caracteres

placeholder: El atributo de placeholder especifica una sugerencia que describe el valor esperado de un campo de entrada (un valor de muestra o una breve descripción del formato). La indicación se muestra en el campo de entrada antes de que el usuario introduzca un valor. Funciona con los siguientes tipos de entrada: texto, búsqueda, url, tel, correo electrónico y contraseña.

value: valor inicial que contiene el control

readonly: solo lectura, no permite edición

disabled: deshabilitado, no se permite edición pero el valor se envía con el formulario

size: El atributo size especifica el tamaño (en caracteres) del campo de entrada.

Atributos nuevos en HTML 5

pattern: El atributo pattern especifica una expresión regular que comprueba el valor

del elemento <input>.

El atributo patrón funciona con los siguientes tipos de entrada: texto, búsqueda, url, tel, correo electrónico y contraseña.

autocomplete

autofocus

form

formaction

formenctype

formmethod

formnovalidate

formtarget

height and width

list

min and max

multiple

pattern (regexp)

placeholder

required

step

HTML - Elemento <input> parte 2

Botón para envío de datos comprendidos entre <form> y</form>

```
<input type="submit" name="Enviar" value="Enviar datos" />
```

Botón para ejecución de acciones en javascript

```
<input type="button" name="alerta" value="Mostrar alerta"  
onclick="alert('Alerta!!!!')" />
```

Otro tipo de botón

```
<button type="button">Click aquí!!!</button>
```

Dentro de un elemento <button> puedes poner contenido, como texto o imágenes. Esta es la diferencia entre este elemento y los botones creados con el elemento <input>. el type puede ser button,reset,submit

HTML - <input> botones de selección y checkbox

- **Radio:** establecer el mismo identificador para el atributo **name** para funcionamiento en conjunto mutuamente excluyente.

```
<input type="radio" name="nacionalidad" value="argentina" checked> Argentina<br><input type="radio" name="nacionalidad" value="uruguay"> Uruguay<br><input type="radio" name="nacionalidad" value="chile"> Chile<br><input type="radio" name="nacionalidad" value="otro"> Otro<br>
```

<input checked="" type="radio"/> Argentina
<input type="radio"/> Uruguay
<input type="radio"/> Chile
<input type="radio"/> Otro

- **Checkbox:** permite seleccionar algún elemento o ninguno

```
<input type="checkbox" name="idioma1" value="ingles" checked> Ingles<br><input type="checkbox" name="idioma2" value="portugues" checked> Portugues<br>
```

Ingles
 Portugues

atributo checked: establece de manera predeterminada la opción seleccionada

HTML - <textarea>

```
<textarea name="observaciones" cols="30" rows="4" placeholder="Ingrese su  
observacion"></textarea>
```

El atributo rows especifica el número visible de líneas en un área de texto.

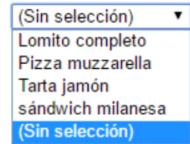
El atributo cols especifica el ancho visible de un área de texto.



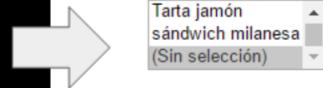
HTML - Elemento <select>

Permite crear un cuadro de lista desplegable o cuadro de lista

```
<select name="menu">
  <option value="lcom">Lomito completo</option>
  <option value="pmuz">Pizza muzzarella</option>
  <option value="tjam">Tarta jam&oacute;n</option>
  <option value="smil">S&aacute;ndwich milanesa</option>
  <option value="" selected>(Sin selecci&oacute;n)</option>
</select>
```



```
<select name="menu" size="3">
  <option value="lcom">Lomito completo</option>
  <option value="pmuz">Pizza muzzarella</option>
  <option value="tjam">Tarta jam&oacute;n</option>
  <option value="smil">S&aacute;ndwich milanesa</option>
  <option value="" selected>(Sin selecci&oacute;n)</option>
</select>
```



El atributo **selected** permite establecer cuál es la opción seleccionada de manera predeterminada.

El atributo **value** es el que valor que va a tomar el elemento <select> cuando se seleccione la opción del cuadro de lista

Otros atributos:

Attribute	Value	Description
autofocus (html5)	autofocus	Specifies that the drop-down list should automatically get focus when the page loads
disabled	disabled	Specifies that a drop-down list should be disabled
disabled		
form (html5)	form_id	Defines one or more forms the select field belongs to
multiple	multiple	Specifies that multiple options can be selected at once
selected at once		
name	name	Defines a name for the drop-down list
required (html5)	required	Specifies that the user is required to select a value before submitting the form
size	number	Defines the number of visible options in a drop-down list

Bibliografía

- <https://www.w3schools.com/html/default.asp>
- <https://developers.google.com/web/>
- [https://developer.mozilla.org/es/docs/HTML/HTML5/Forms_in_H
TML5](https://developer.mozilla.org/es/docs/HTML/HTML5/Forms_in_HTML5)
- <https://www.w3.org/TR/html5/Overview.html>