

# UT0 INTRODUCCIÓN A LAS ARQUITECTURAS Y HERRAMIENTAS DE PROGRAMACIÓN WEB

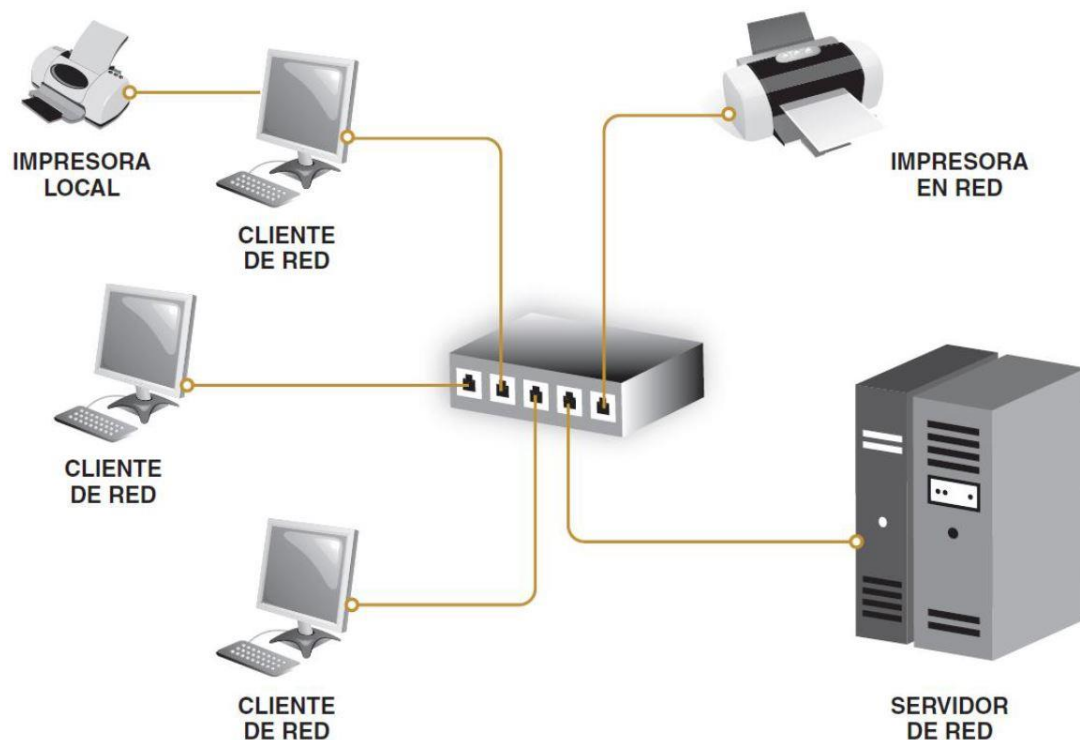
## 0.1.- EVOLUCIÓN Y CARACTERÍSTICAS DE LOS NAVEGADORES WEB

La “World Wide Web” (Web) es una red de ordenadores internacional que permite el intercambio de información entre particulares y empresas, estados, y todas las combinaciones que se ocurran.

El funcionamiento de la web es posible debido a (Niveles OSI):

1. en primer lugar, a una infraestructura Hardware (cables, repetidores, puentes, pasarelas, enrutadores, etc.),
2. en segundo lugar, a componentes Software
  - A. protocolos de comunicaciones (TCP, IP, HTTP, FTP, SMTP, etc.),
  - B. Sistema de nombres de dominio (DNS: permite la búsqueda y recuperación de recursos)
  - C. software específico para proveer y consumir dichos recursos.

En este contexto, la configuración arquitectónica más habitual se basa en la estructura Cliente-servidor:



Esta arquitectura está basada en la idea de servicio, en el que el cliente (S.O. Monousuario) es un componente consumidor de servicios y el servidor (S.O. Multiusuario) es un proceso proveedor de servicios. La relación existente entre ambos se establece mediante el intercambio de mensajes que permiten el acoplamiento entre ambos.

En cuanto a los componentes software del cliente, el más habitual e importante es el **NAVEGADOR WEB (BROWSER)**, cuya **función básica** es **permitir al cliente conectarse al servidor para acceder a su información**. Se distribuyen habitualmente como software libre y **permiten acceder y visualizar un recurso publicado por un servidor web a través de Internet y descrito mediante una dirección URL (Universal Resource Locator)**.

El uso más habitual del Navegador web es para “explorar” recursos de tipo hipertexto, comúnmente descritos en HTML, ofrecidos por servidores web de todo el mundo a través de internet.

Los exploradores o navegadores han ido evolucionando de manera que configuraban al cliente menos o más capacidades:

- Navegadores que sólo permiten conectarse al servidor.
- Navegadores que permiten conectarse al servidor, procesar datos, sincronizar datos sin intervención del usuario. Esto es gracias a lenguajes como Java, DHTML, etc..

Desde la creación de la web a principios de los años 90, los navegadores web han evolucionado desde meros visualizadores de texto hasta los actuales navegadores, totalmente preparados para soportar cualquier tipo de interacción y funcionalidad requerida por el usuario.

Lista evolutiva de los navegadores más relevantes:

- ❖ **Mosaic:** Se considera uno de los primeros navegadores web.
  - el primero con capacidades gráficas.
  - Las primeras versiones se diseñaron para ser ejecutado sobre UNIX pero, debido a su gran aceptación, pronto fue portado a las plataformas de Windows y Macintosh.
  - Se utilizó como base para las primeras versiones de Internet Explorer y Mozilla. Su desarrollo se abandona en 1997.
- ❖ **Netscape Navigator:** (después Communicator),
  - fue el primer navegador en incluir un módulo para la ejecución de código *script* (JavaScript).
  - Se le considera el perdedor de la “Guerra de los navegadores” que tuvo lugar entre Netscape y Microsoft por el dominio del mercado de navegadores web a finales de los años 90.
  - Sus características, sin embargo, se consideran la base de otros navegadores, como Mozilla Firefox.
- ❖ **Internet Explorer (Edge a partir de Windows 10):** Navegador de Microsoft.
  - Su cuota de distribución y uso ha sido muy elevada gracias a su integración con los sistemas Windows.
  - En los últimos años su utilización ha ido descendiendo paulatinamente.

- Este navegador no ha dejado de evolucionar incorporando en cada versión nuevos estándares web, personalización de la navegación, seguridad, etc..
- ❖ **Mozilla Firefox:**
  - Se trata de un **navegador de código abierto multiplataforma** de gran aceptación en la comunidad de desarrolladores web.
  - Existen gran variedad de utilidades, extensiones y herramientas que permiten la personalización tanto del funcionamiento del navegador como de su apariencia.
  - Fue uno **de los primeros en incluir la navegación por pestañas**.
  - Además, es un navegador multiplataforma, lo que le ha llevado a poder recortar parte de la cuota de distribución que desde los inicios del 2000 venía teniendo Internet Explorer.
- ❖ **Google Chrome:** Desde septiembre de 2008 es el navegador de Google.
  - **compilado a partir de componentes de código abierto.**
  - Tiene como características principales la seguridad, velocidad y estabilidad.
  - En numerosos test comparativos este navegador ha demostrado ser **uno de los más rápidos y seguros** gracias,
    - ✓ entre otras razones, a **estar construido siguiendo una estructura multiproceso** en la que cada pestaña es ejecutada de forma independiente.
- ❖ **Safari:** Navegador por defecto de los **sistemas Apple**, aunque **también** se han desarrollado versiones para funcionamiento **en las plataformas Windows**.
  - Las últimas versiones incorporan las características habituales:
    - ✓ de navegación por pestañas,
    - ✓ corrector ortográfico en formularios,
    - ✓ almacenamiento de direcciones favoritas (“marcadores”),
    - ✓ bloqueador de ventanas emergentes,
    - ✓ soporte para motores de búsqueda personalizados
    - ✓ un gestor de descargas propio
    - ✓ opciones de privacidad, iCloud, etc..
- ❖ **Dolphin Browser:** Debido al auge de los dispositivos móviles inteligentes (smartphones y tablets) y de los sistemas operativos para estos, tenemos que hacer referencia obligatoriamente a uno de los navegadores más populares para estas plataformas. **Específico para el sistema operativo Android**,
  - fue uno **de los primeros en incluir soporte para navegación multitáctil**.
  - Adblock (bloqueo de anuncios), navegación de forma Incógnita y Flash.
  - Utiliza un motor de renderizado de páginas similar al de Chrome o Safari.

**Parámetros que permiten diferenciar los navegadores** descritos anteriormente:

1. **Plataforma de ejecución:** Sistemas operativos sobre los que se pueden ejecutar.
2. **Características del navegador:** La mayoría de los navegadores ofrecen funcionalidades adicionales asociadas a la experiencia del usuario a la hora de navegar por la Red.

Alguna de las **características soportadas de forma Nativa** (es decir, **sin necesidad de instalar extensiones**) incluyen:

- la administración de marcadores,
  - gestores de descarga,
  - almacenamiento seguro de contraseñas y datos de formularios,
  - corrección ortográfica o
  - definición de herramientas de búsqueda.
3. **Personalización de la Interfaz:** Las **funciones de accesibilidad** que definen la experiencia del usuario con un navegador web también son su aspecto diferencial.  
Entre los aspectos más destacados podemos mencionar:
    - el soporte para la navegación por pestañas,
    - la existencia de los bloqueadores de ventanas emergentes,
    - la integración de visualizadores de formatos de ficheros (como PDF),
    - opciones de zoom o
    - funciones avanzadas de búsqueda de texto en páginas web.
  4. **Soporte de tecnologías Web:** Actualmente, una de las mayores preocupaciones de los desarrolladores de navegadores web es el grado de soporte de los estándares de la web.

Por ello, **podemos clasificar los navegadores de acuerdo a su nivel de soporte de tecnologías** como:

- CSS (hojas de estilo en cascada),
  - Java,
  - lenguajes de scripting del cliente (JavaScript),
  - **RSS (Really Simple Syndication)**, un formato XML para syndicar o compartir contenido en la web) o
  - Atom (sindicación de contenidos),
  - XHTML (HTML con formato de XML), etc..
5. **Licencia de Software:** Existen navegadores:
    - de código libre, como:
      - ✓ Mozilla Firefox (licencia GNU GPL) o
      - ✓ Google Chrome (licencia BSD), y
    - navegadores propietarios como:
      - ✓ Edge (Microsoft) o
      - ✓ Safari (Apple).

Salvo raras excepciones (OmniWeb) todos los navegadores son gratuitos.

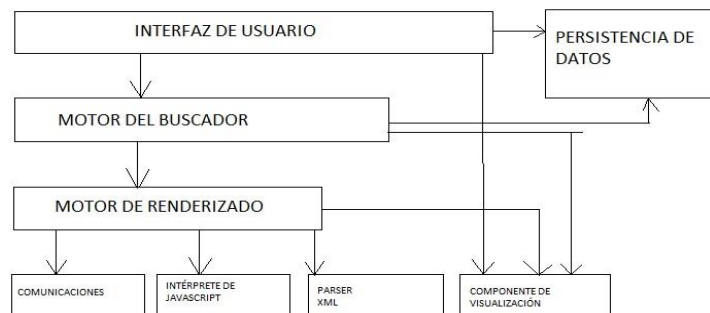
## 0.2.- DIAGRAMA DE BLOQUES DE UN NAVEGADOR. SUS CAPAS.

Cada navegador web tiene su propia forma de interpretar la interacción con el usuario. El resultado de **esta interacción**, en cualquier caso, **se inicia con el usuario indicando la dirección del recurso al que quiere acceder y termina con la visualización del recurso por parte del navegador en la pantalla del usuario** (salvo interacciones posteriores del usuario con la página). La forma de realizar este proceso depende del propósito del navegador y de la configuración del mismo. De esta forma, un navegador puede estar más centrado en:

- ❖ ofrecer una respuesta más rápida,
- ❖ en mostrar una respuesta más fiel al contenido del recurso obtenido,
- ❖ en priorizar los aspectos de seguridad de las comunicaciones con el servidor, etc..

Para poder llevar a cabo el proceso descrito anteriormente, cada navegador está formado por una serie de elementos y componentes determinados que conforman lo que se denomina la *Arquitectura del Navegador*. A pesar de que **cada navegador tiene su propia arquitectura**, la gran mayoría de ellos coinciden en una serie de componentes básicos y comunes en todos ellos, es lo que llamamos *arquitectura de referencia*.

De acuerdo al estudio llevado a cabo por **Grosskurth y Godfrey** los **componentes básicos** incluidos **en la arquitectura de referencia de un navegador web** son los reflejados en la siguiente figura:



Los componentes de esta arquitectura de referencia son:

1. **Subsistema de interfaz de usuario:** Es la capa que actúa de intermediario, interfaz, entre el usuario y el motor del buscador (o de navegación).  
¿Qué funcionalidades ofrece?
  - la visualización de la barra de herramientas,
  - progreso de carga de la página,
  - gestión inteligente de las descargas,
  - preferencias de configuración de usuario y de impresión.
  - En algunos casos puede comunicarse con el sistema operativo para el manejo de sesiones de usuario o el almacenamiento de preferencias de visualización o configuración.

2. **Subsistema del motor del buscador o motor de navegación:** Este subsistema es un componente que ofrece una interfaz de alto nivel para el motor de renderizado.
- ¿Cuáles son sus funciones principales?
- cargar una dirección determinada (URL o URI)
  - soportar los mecanismos básicos de navegación tales como ir a la página anterior o siguiente o la recarga de la página.
  - gestionar las alertas de JavaScript (mensajes de usuario)
  - gestionar el proceso de carga de una página (es quien le provee de información a la interfaz de usuario al respecto).
  - Y, finalmente, es el encargado de consultar y administrar las preferencias de ejecución del motor de renderizado.
3. **Subsistema de renderizado:** Este componente es el encargado de producir una representación visual del recurso obtenido a partir del acceso a un recurso web.
- ¿Cuáles son las funciones de este módulo?
- **Interpretar el código** de la página web a la que se accede. En función de los lenguajes, estándares y tecnologías soportadas por el navegador, este módulo será capaz de mostrar documentos HTML, XML, hojas de estilo CSS e incluso contenido embebido en la página (audio/vídeo) e imágenes.
  - **Dimensionar** de manera exacta cada elemento a mostrar y, en ocasiones, **posicionar** dichos elementos en una página.
- Algunos de los motores de renderizado más utilizados son:
- *Gecko*, utilizado en Firefox, Mozilla Suite y otros navegadores como Galeon.
  - *Trident*, el motor de Internet Explorer para Windows (EdgeHTML para Edge).
  - *WebKit*, el motor de Google Chrome, Epíphany y Safari.
  - *Presto*, el motor de Opera.
  - *Tasman*, el motor de Internet Explorer para Mac.
4. **Subsistema de comunicaciones:** ¿Cuáles son sus funciones?
- **Implementar** los **protocolos** de transferencia de ficheros y documentos utilizados en Internet (HTTP, FTP, etc.).
  - **Identificar la codificación** de los datos obtenidos en función de su tipo, de tal forma que es capaz de identificar si el recurso obtenido es de tipo texto, audio, vídeo, etc. (codificado en estándar MIME, *Multipurpose Internet Mail Extensions*).
  - **Almacenar una caché de elementos accedidos recientemente**, esto dependerá de las capacidades personalizadas para el navegador.
5. **Intérprete de JavaScript:** Las páginas HTML llevan código intercalado para la provisión de ciertas funcionalidades al usuario como puede ser la respuesta a ciertos eventos del teclado o del ratón. El lenguaje comúnmente aceptado para la programación de este código embebido es JavaScript (siguiendo el estándar ECMAScript).

Por tanto, ¿Cuál es la función de este módulo?

- Analizar y ejecutar código JavaScript.

Este módulo **puede ser configurado, e incluso deshabilitado**, por cuestiones de seguridad o facilidad de navegación **desde el motor de navegación o el motor de renderizado** (por ejemplo, para evitar que aparezcan ventanas emergentes).

La existencia de módulos de interpretación de código difiere de un navegador a otro. Por ello, es posible que existan subsistemas intérpretes de otros lenguajes, como *applets* de Java, AJAX o ActionScript.

#### 6. **Parser XML:**

¿Cuál es la función de este módulo (*parser*)?

- Permitir **cargar en memoria** una representación en árbol (árbol DOM, *Document Object Model*) de la página.
- Así, poder acceder más fácilmente a los contenidos definidos en un documento HTML (en realidad XHTML).
- En consecuencia, un **acceso mucho más rápido** a los diferentes elementos de una página por parte del navegador.

#### 7. **Componente de visualización:** Este subsistema ofrece funcionalidades relacionadas con la visualización de los contenidos de un documento HTML en una página web.

Ofrece a los subsistemas principales del navegador web

- primitivas de dibujo y posicionamiento en una ventana.
- un conjunto de componentes visuales predefinidos (*widgets*) y
- un conjunto de fuentes tipográficas.

Suele estar muy relacionado con las librerías de visualización del Sistema operativo.

#### 8. **Subsistema de persistencia de datos:** Funciona como almacén de diferentes tipos de datos para los principales subsistemas del navegador. Estos datos, dependiendo del nivel pueden ser:

- A bajo nivel, este sistema administra
  - certificados de seguridad y
  - las cookies.
- A alto nivel, suelen estar relacionados con el
  - almacenamiento de historiales de navegación
  - mantenimiento de sesiones de usuario en disco.
  - Preferencias de configuración del navegador (de barra de herramientas, por ejemplo)
  - el listado de marcadores.

### 0.3.- LENGUAJES Y TECNOLOGÍAS DE PROGRAMACIÓN EN ENTORNO CLIENTE.

Los lenguajes de programación del entorno de cliente **son aquellos que se ejecutan en el navegador web**, dicho de otro modo, **en el lado del cliente** dentro de una estructura cliente/servidor.

- El lenguaje cliente principal es HTML (lenguaje de marcado de hipertexto, *Hypertext Markup Language*), ya que la mayoría de páginas en el servidor son codificadas siguiendo este lenguaje para describir la estructura y el contenido de una página en forma de texto.  
Existen algunas alternativas y variaciones de este lenguaje tales como:
  - XML (Lenguaje de marcas extensible, *eXtensible Markup Language*).
  - DHTML (Dynamic HTML)
  - XHTML (eXtensible HTML).
- Con el fin de mejorar la interactividad con el usuario, en este grupo de lenguajes cliente podemos incluir todos los lenguajes *script*, tales como:
  - JavaScript (el más utilizado dentro de esta rama)
  - VBScript.
- También existen otros lenguajes más independientes:
  - ActionScript (para crear contenido Flash).
  - AJAX (como extensión de JavaScript para comunicación Asíncrona).

A continuación vamos a analizar brevemente estos lenguajes.

#### 0.3.1 HTML Y DERIVADOS

HTML es una particularización de un lenguaje anterior, SGML (Standard Generalized Markup Language), un sistema para la organización y etiquetado de documentos estandarizado en 1986 por la organización ISO. El término HTML se suele referir a ambas cosas, tanto al tipo de documento como al lenguaje de marcas.

El HyperText Markup Language (Lenguaje de marcado de hipertexto) es el lenguaje de marcas de texto más utilizado en la *world wide web*. Fue creado en 1989 por Tim Berners-Lee a partir de dos elementos previos para crear dicho lenguaje:

- ✚ por un lado, el concepto de hipertexto como herramienta básica para conectar dos elementos (documentos o recursos) entre sí;
- ✚ SGML, como lenguaje básico para colocar etiquetas o marcas en un texto.

Debemos tener en cuenta que **HTML no es propiamente un lenguaje de programación** como puede serlo Java o VisualBasic, **sino que se basa en la utilización de un sistema de etiquetas cerrado aplicado a un documento de texto**. Además, este lenguaje **no necesita ser compilado**, sino que **es interpretado** (ejecutado a medida que se avanza en el documento HTML). Una característica particular de HTML es que, ante algún error de sintaxis que presente el texto, HTML no lo detectará y seguirá con la interpretación del siguiente fragmento



del documento. **El entorno para desarrollar HTML puede ser simplemente un procesador de textos.**

Con el lenguaje HTML se pueden hacer gran variedad de acciones:

- ✚ organizar simplemente el texto y los objetos de una página web,
- ✚ crear listas y tablas,
- ✚ la esencia de la web: los hipervínculos. Un hipervínculo es un enlace de una página web o un archivo a otra página web u otro archivo. Cuando un usuario hace clic en el hipervínculo, el destino se mostrará en un explorador web, se abrirá o se ejecutará, en función del tipo de recurso destino. El destino es con frecuencia otra página web, pero también puede ser una imagen, un archivo multimedia, un documento de MicroSoft Office, una dirección de correo electrónico, un programa, etc..

Con el tiempo, HTML ha ido evolucionando, dando lugar a lenguajes derivados de éste, como XML, XHTML o DHTML, en función de la flexibilidad ofrecida al conjunto de etiquetas admitido o de la integración de HTML con otros lenguajes que permitan dotar de más dinamismo e interactividad a las páginas creadas con HTML.

✚ XML es un lenguaje de etiquetado extensible muy simple, pero con unas reglas de codificación muy estrictas que juegan un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML (también deriva de SGML), pero cuyo **objetivo principal es describir datos para su transferencia eficiente y no mostrarlos**, como es el caso de HTML.

✚ XHTML, es también muy similar a HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje XML. Técnicamente, como ya se ha comentado, HTML es descendiente directo del lenguaje SGML, mientras que XHTML lo es del XML (que, a su vez, también es descendiente directo de SGML). Las páginas y documentos creados con XHTML son muy similares a las páginas y documentos HTML. Actualmente, los procesos de estandarización de HTML y XHTML siguen procesos paralelos.

✚ DHTML (el HTML dinámico) que consiste en una forma de aportar interactividad a las páginas web. El origen de DHTML hay que buscarlo en la versión 4.0 de los navegadores de NetScape Communicator e Internet Explorer (y posteriores versiones de ambos navegadores), que **permitieron la integración de HTML con lenguajes de *scripting*** (JavaScript), **hojas de estilo personalizadas** (CSS) y la **identificación de los contenidos de una página web en formato de árbol** (DOM). Es la combinación de estas tecnologías la que permite aumentar la funcionalidad e interactividad de la página.

La **principal aportación de DHTML** es servir de herramienta con la que podemos crear efectos **que requieren poco o ningún ancho de banda ya que se ejecutan en el entorno del cliente.** Se puede utilizar para crear animaciones, juegos, aplicaciones, etc., para introducir nuevas formas de navegar a través de los sitios web y para crear un auténtico entramado de capas que solo con HTML sería imposible de abordar. Aunque muchas de las características del DHTML se podrían duplicar con otras herramientas como Java o Flash, **DHTML ofrece la ventaja de que no requiere ningún tipo de extensión para poderlo utilizar.**




Podríamos decir que HTML es el lenguaje más importante en el ámbito de la world wide web puesto que casi todos los lenguajes utilizados en la web, de acuerdo a la tendencia mostrada en la última década, terminan confluyendo hacia una representación en HTML para que nuestro navegador lo pueda leer y visualizar en la pantalla del cliente.

### 0.3.2 CSS

Las CSS (Hojas de Estilo en Cascada, Cascade Style Sheets) **sirven para separar el formato que se quiere dar a la página web de la estructura de la página web y las demás instrucciones.** Utilizamos CSS, por ejemplo, cuando queremos que en determinados párrafos de nuestra página web se use un determinado tipo y tamaño de letra, un color de fuente y un color de fondo. En vez de tener que definir párrafo por párrafo todos los atributos del formato que queremos dar, solo hace falta que lo definamos una vez, en la hoja de estilo (CSS). Nos basta con poner una referencia en nuestro documento HTML que la dirija al formato que queremos darle, definido en la hoja de estilo. De esta forma, solo debemos poner esa referencia en cada párrafo, en vez de especificar el formato uno por uno.

### 0.3.3 JAVASCRIPT

JavaScript es un lenguaje de programación de *scripting* (interpretado) y, normalmente, embebido en un documento HTML. Se define como:

-  orientado a objetos,
-  débilmente tipado
-  y con características dinámicas

Se utiliza principalmente su forma del lado del cliente, con un intérprete implementado como parte de un navegador web. Su **objetivo principal** es el de permitir **realizar mejoras en la interfaz de usuario** y, de esta forma, **crear páginas web dinámicas**. Existe, no obstante, una forma de JavaScript del lado del servidor.

Inicialmente, se diseñó con una sintaxis similar al lenguaje C, aunque adopta nombres y convenciones propias del lenguaje de programación Java. Sin embargo, tenemos que dejar claro que **Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes**. Actualmente,

existen dos especificaciones estándares denominadas ECMAScript e ISO/IEC 16262, que son generalizaciones del lenguaje JavaScript (que se creó antes que los estándares). A partir de la versión 5.1 de ECMAScript, este lenguaje está totalmente alineado con el estándar ISO 16262. JavaScript, y otros lenguajes similares como ActionScript para Adobe Flash, se considerarán dialectos del estándar.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Ya que este lenguaje es uno de los más predominantes entre los lenguajes dedicados al cliente web, hablaremos sobre su sintaxis y uso a lo largo del curso.

### 0.3.4 APPLETS DE JAVA

Una manera de incluir funcionalidades complejas en el ámbito de una página web se puede hacer **integrando pequeños componentes** (**objetos independientes**) en dicha página.

Cuando la tecnología utilizada en estos componentes es **Java** lo denominamos **applets** (**es importante que tengamos en cuenta que estos fragmentos de código Java se ejecutan en el cliente**). Estos *applets* se programan en Java y, por tanto, se benefician de la potencia y la flexibilidad que este lenguaje nos ofrece. Es otra manera de incluir código para ejecutar en los clientes que visualizan una página web. Se trata de pequeños programas que se transfieren con la página web y que el navegador ejecuta en el espacio de la página (a través de un módulo o extensión concretos).

Características de los *applets*:

- ✓ se programan en Java
- ✓ se envían al cliente **precompilados**, es por ello que la manera de trabajar de estos varía un poco respecto a los lenguajes de *script* como JavaScript.

**Ventajas** de los *applets*:

- ✓ son mucho menos dependientes del navegador que los scripts en JavaScript.
- ✓ Son independientes del sistema operativo del ordenador donde se ejecutan.
- ✓ Al ser Java más potente que JavaScript, el número de aplicaciones de los *applets* podrá ser mayor.

**Desventajas** de los *applets* frente al uso de JavaScript:

- ✓ Los *applets* son más lentos de procesar.
- ✓ Tienen un espacio delimitado en la página donde se ejecutan, es decir, **no se mezclan con todos los componentes de la página ni tienen acceso a ellos**.

- ✓ Consecuencia de lo anterior, con los *applets* **no se podrán realizar acciones como:**
  - abrir ventanas secundarias,
  - controlar marcos (frames),
  - obtener la información de formularios,
  - administrar capas, etc..

### 0.3.5. AJAX

AJAX, acrónimo de *Asynchronous JavaScript And XML* (JavaScript Asíncrono y XML), es un conjunto de técnicas y métodos de desarrollo web para la creación de aplicaciones web interactivas.

Características principales de AJAX:

- ✓ Este tipo de aplicaciones se ejecutan en el lado del cliente, es decir, en el navegador de los usuarios que acceden a una página web.
- ✓ Al contrario que una página web **HTML/XHTML/DHTML**, en la que la **comunicación se interrumpe** una vez el cliente recibe la página, con **AJAX** se mantiene la **comunicación asíncrona con el servidor en segundo plano** (sin que el usuario sea consciente de dicha comunicación).
- ✓ Consecuencia directa de lo anterior es que se podrán realizar cambios sobre las páginas del cliente sin necesidad de que éste proceda a recargarlas.
- ✓ Mayor interactividad con el usuario
- ✓ Mayor velocidad en las aplicaciones.

El fundamento de AJAX se encuentra en la utilización de un objeto específico de JavaScript denominado *XMLHttpRequest*, disponible y aceptado por la mayoría de los navegadores actuales. AJAX no es una tecnología en sí misma, sino que en realidad es una combinación de 4 tecnologías existentes:

1. Lenguaje **XHTML/HTML** y hojas de estilo en cascada (**CSS**), con los que se especifican la estructura y contenidos de la página web.
2. **DOM**, como forma de organizar en árbol los contenidos de una página para así poder acceder más fácilmente a un elemento determinado.
3. El objeto *XMLHttpRequest*, que es el que tiene implementadas las operaciones necesarias para comunicarse asíncronamente con el servidor.
4. **XML**, como lenguaje utilizado por el objeto *XMLHttpRequest* para recuperar e intercambiar información con el servidor (aunque cualquier formato es válido).

Como podemos ver, **AJAX** está basado en estándares abiertos y ampliamente soportados, como JavaScript o DOM, por lo que **es válida para múltiples plataformas** y utilizable en **muchos sistemas operativos y navegadores**.

Como **desventajas de AJAX** cabe mencionar:

- ✓ La curva de aprendizaje del desarrollo de este tipo de aplicaciones es más elevada.
- ✓ El navegador no es capaz de guardar un historial real de los contenidos a los que se ha accedido (debido a la comunicación asíncrona).

### 0.3.6 ADOBE FLASH Y ACTIONSCRIPT

Flash es una tecnología de animación actualmente bajo licencia de Adobe (propietario) y que utiliza ActionScript como lenguaje principal. La principal ventaja de Flash es la capacidad para crear gráficos y animaciones vectoriales.

La evolución de **Flash** lo ha llevado de ser una simple herramienta de animación y dibujo de escritorio a ser una base para la programación multimedia y, más recientemente, convertirse en una completa aplicación de desarrollo para la web. **Posee su propio lenguaje de programación orientado a objetos derivado del estándar ECMAScript (denominado ActionScript) que puede adaptarse a la mayoría de los navegadores y sistemas operativos.** En los últimos años, la tendencia muestra que la tecnología Flash, por los motivos que se verán más adelante, está en claro declive, pudiéndose observar una mayor prevalencia de otras tecnologías y lenguajes como HTML5.

Esta circunstancia adversa no es inconveniente para que entendamos que las posibilidades de Flash son extraordinarias, puesto que cada nueva versión ha mejorado a la anterior. Aunque su uso más frecuente es el de crear animaciones sus usos pueden ser otros (reproducción de vídeo, por ejemplo). **El uso de Flash ha permitido crear aplicaciones interactivas de gran complejidad y visualmente muy atractivas que no sería posible crear utilizando DHTML o XHTML directamente.** Además, gracias a las características dinámicas del lenguaje utilizado, la utilización de Flash permite aumentar el grado de interactividad del usuario con la página web.

En el lado opuesto, entre las **desventajas** que tradicionalmente se le han detectado al desarrollo de aplicaciones y páginas web basadas en esta tecnología podemos destacar, entre otras, el hecho de que,

- **al tratarse de creación de animaciones de índole vectorial, el consumo de procesador (y de batería en dispositivos móviles) es más elevado**
- se trata de un software 100% propietario, coartando la libertad de extender o mejorar el núcleo de Flash.

El que Flash sea un programa de animación vectorial significa que se pueden crear animaciones complejas: aumentar y reducir elementos de la animación, mover de posición estos objetos, y otras cosas **sin que la animación ocupe mucho espacio en el disco**. Los vectores con los que trabaja Flash son, por decirlo de alguna manera, **siluetas** que casi no ocupan espacio y se pueden modificar fácilmente y sin gasto de memoria en disco.

¿Qué es Animación? Es el proceso que se utiliza para crear una sensación de movimiento a imágenes o dibujos. Existen muchos tipos de animación desde los dibujos animados, la animación de objetos, muñecos, marionetas, figuras de plastilina, maquetas de modelos a escala, objetos comunes hasta personas.

Animación vectorial: La animación vectorial, está compuesta por el movimiento de una o más formas en pantalla con la diferencia primordial de que dicha animación está compuesta por diferentes segmentos.

## 0.4 INTEGRACIÓN DEL CÓDIGO CON LAS ETIQUETAS HTML.

Para que podamos desarrollar aplicaciones web que se ejecuten en el lado del cliente lo primero que debemos saber es que el documento base estará escrito en HTML (o XHTML en su defecto). En este punto nos centraremos en algunas de las formas que un desarrollador de páginas web tiene a su disposición a la hora de integrar código de scripting en documentos HTML.

La integración de JavaScript y HTML/XHTML, podemos hacerla de diferentes formas:

- ❖ En el mismo documento.
- ❖ En archivo externo.
- ❖ En elementos HTML

### 0.4.1.- JAVASCRIPT EN EL MISMO DOCUMENTO HTML.

HTML se basa en el uso de etiquetas predefinidas para marcar el texto. Una de estas etiquetas es `<script>`, para el inicio de código JavaScript y `</script>` para la finalización de ese código embebido. Esta etiqueta puede incluirse en cualquier parte del documento, aunque **se recomienda** que el código JavaScript, salvo para propósitos concretos de generación de contenidos a visualizar, **se defina dentro de la cabecera** del documento HTML (entre las etiquetas `<head>` y `</head>`). Podemos ver un ejemplo a continuación:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title>Ejemplo 1</title>
    <script type="text/javascript">
      alert("Prueba de JavaScript");
    </script>
  </head>
  <body>
    <h1>Ejemplo 1: Código embebido</h1>
  </body>
</html>
```

El ejemplo anterior muestra una página XHTML. Para que sea válida tenemos que incluir la cabecera “<DOCTYPE....” y añadir el atributo *type* con el valor correcto a la etiqueta <script>. Los valores válidos para este atributo están estandarizados. Para el caso de JavaScript, el valor correcto es *text/javascript*.

Esta técnica suele utilizarse cuando se definen instrucciones que se referenciarán desde cualquier parte del documento o cuando se definen funciones con fragmentos de código genéricos. La mayor desventaja de este método es que, si ese fragmento de código lo queremos utilizar en otras páginas, debemos incluirlo en cada una de ellas, lo cual es un inconveniente cuando tenemos que realizar modificaciones de dicho código.

#### 0.4.2.- JAVASCRIPT EN UN ARCHIVO EXTERNO

Las mismas instrucciones de JavaScript que se incluyen entre un bloque <script> </script> pueden almacenarse en un fichero externo con extensión .js. Al igual que sucede con los documentos HTML, los ficheros .js pueden crearse con cualquier editor de texto. A continuación se muestra el contenido de un fichero externo que contiene código JavaScript.

```
Archivo mensaje.js
alert("Prueba de JavaScript");
```

La forma de acceder y enlazar esos ficheros .js con el documento HTML/XHTML es a través de la propia etiqueta <script>. No existe un límite en el número de ficheros .js que pueden enlazarse en un mismo documento HTML/XHTML. El siguiente ejemplo muestra cómo se enlazaría un documento HTML/XHTML con el fichero anterior *mensaje.js*:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo 1</title>
<script type="text/javascript" src="mensaje.js">
</script>
</head>
<body>
<h1>Ejemplo 1: Código embebido</h1>
</body>
</html>
```

Al igual que en el ejemplo anterior, para que esta técnica funcione, además del atributo *type* tenemos que incluir el atributo *src*. Este atributo contendrá un valor que indicará la ruta relativa (con respecto al fichero HTML/XHTML) de la ruta donde se encuentra el archivo JavaScript que se quiere enlazar. En este caso está dentro de la misma carpeta. La única restricción de la etiqueta <script> es que solo puede enlazarse un archivo en cada etiqueta. Podemos incluir el número de etiquetas <script> que necesitemos.

Entre las ventajas de este método está que la vinculación de un mismo fichero externo puede hacerse desde varios documentos HTML/XHTML distintos. De esta forma, en el caso que haya que modificar algo, solo hay que hacerlo en un único fichero. Cualquier modificación realizada en el fichero externo se ve reflejada inmediatamente en todas las páginas que lo enlacen.

#### **0.4.3.- JAVASCRIPT EN ELEMENTOS HTML**

El último método suele utilizarse habitualmente como forma de controlar los eventos que suceden asociados a un elemento HTML concreto (aunque también puede utilizarse con otros fines). Consiste en insertar fragmentos de JavaScript dentro de atributos de etiquetas HTML de la página:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1-  
transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />  
<title>Ejemplo 1</title>  
</head>  
<body>  
<p onclick="alert('Prueba de JavaScript');">  
    Ejemplo 1: Código embebido en atributos  
</p>  
  
</body>  
</html>
```

La principal desventaja es que el código JavaScript se intercala con el código HTML y, dependiendo de la complejidad y longitud de éste, el mantenimiento y modificación del código puede resultar más complicado.