UD 1

SELECCIÓN DE ARQUITECTURAS Y HERRAMIENTAS DE PROGRAMACIÓN.

1.- OBJETIVOS.

- Conocer las diferentes alternativas existentes para la navegación web en función de las diferentes tecnologías web que se ejecutan en un cliente.
- Reconocer las capacidades de la ejecución de código en el lado del cliente de acuerdo a los componentes arquitectónicos de un navegador web.
- Identificar los principales lenguajes y tecnologías de programación en entorno cliente.
- Conocer las técnicas de integración del código con documentos HTML

2.- EVOLUCIÓN Y CARACTERÍSTICAS DE LOS NAVEGADORES WEB.

Para realizar cualquier desarrollo web, es imprescindible comprobar que el resultado que queremos es el adecuado con la mayor cantidad de navegadores posibles, especialmente aquellos más usados.

Además de procesar etiquetas HTML, los navegadores suelen interpretar lenguajes de script, siendo JavaScript uno de los más populares.

Vamos a comenzar viendo como ha sido la evolución, así como las características de los navegadores.

La Web o World Wide Web representa un universo de información accesible globalmente a través de Internet. Está formado por un conjunto de recursos interconectados que conforman el conocimiento humano actual. El funciónenlo de la Web es posible debido a la coexistencia de una serie de componentes software y hardware. Estos elementos abarcan desde los componentes físicos de Internet (hub, repetidores, switch, puentes, pasarelas, etc.) y los protocolos de comunicación (TCP, IP, HTTP, FTP, SMTP, etc.) hasta la utilización de nombres de dominio.

En este contexto, el desarrollo en entornos web debe tener en cuenta la distribución de los elementos y la función que tiene cada uno de ellos. La configuración arquitectónica más habitual se basa en el modelo denominado Cliente/Servidor, basado en la idea de servicio, en el que el cliente es un componente consumidor de servicios y el servidor es un proceso proveedor de

servicios. Además, esta relación esta robustamente comentada en el intercambio de mensajes como único elemento de acoplamiento entre ambos.

2.1.- Definición de navegador.

Un navegador o navegador web, o browser (en inglés), es un software que permite el acceso a Internet, interpretando la información de distintos tipos de archivos y sitios web para que estos puedan ser visualizados.

La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados. Además, permite visitar páginas web y hacer actividades en ella, es decir, podemos enlazar un sitio con otro, imprimir, enviar y recibir correo, entre otras funcionalidades más.

Los documentos que se muestran en un navegador pueden estar ubicados en el ordenador en donde está el usuario, pero también pueden estar en cualquier otro dispositivo que esté conectado en el ordenador del usuario o a través de Internet, y que tenga los recursos necesarios para la transmisión de los documentos (un software servidor web).

Tales documentos, comúnmente denominados páginas web, poseen hipervínculos que enlazan una porción de texto o una imagen a otro documento, normalmente relacionado con el texto o la imagen.

El seguimiento de enlaces de una página a otra, ubicada en cualquier ordenador conectado a Internet, se llama navegación, de donde se origina el nombre navegador.

Para acceder a estos recursos, se utiliza un identificador único llamado URL (Uniform Resource Locator).

El formato general de una URL es “protocolo://máquina/directorio/archivo”.

- Si no se especifica el directorio, toma como directorio el raíz.
- Si no se especifica el fichero, toma alguno de los nombres por defecto (“index.html”, “index.php”, etc...)

Un ejemplo muy típico es: http://www.google.es donde se accede al recurso www.google.es usando el protocolo https.

2.2.- Funcionamiento de los navegadores.

La comunicación entre el servidor web y el navegador se realiza mediante el protocolo HTTP, aunque la mayoría de los navegadores soportan otros protocolos como FTP y HTTPS (una versión cifrada de HTTP basada en Secure Socket Layer o Capa de Conexión Segura (SSL)).

La función principal del navegador es obtener documentos HTML e interpretarlos para mostrarlos en pantalla. En la actualidad, no solamente descargan este tipo de documentos sino que muestran con el documento sus imágenes, sonidos e incluso vídeos streaming en diferentes

formatos y protocolos. Además, permiten almacenar la información en el disco o crear marcadores (bookmarks) de las páginas más visitadas.

Algunos de los navegadores web más populares se incluyen en lo que se denomina una Suite. Estas Suite disponen de varios programas integrados para leer noticias de Usenet y correo electrónico mediante los protocolos NNTP, IMAP y POP.

Los primeros navegadores web sólo soportaban una versión muy simple de HTML. El rápido desarrollo de los navegadores web propietarios condujo al desarrollo de dialectos no estándares de HTML y a problemas de interoperabilidad en la web. Los más modernos (como Google Chrome, Mozilla, Netscape, Opera e Internet Explorer / Microsoft Edge) soportan los estándares HTML y XHTML (comenzando con HTML 4.01, los cuales deberían visualizarse de la misma manera en todos ellos).

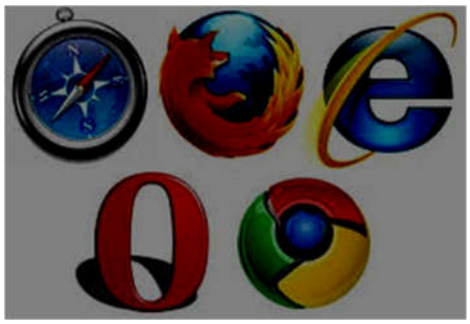
Los estándares web son un conjunto de recomendaciones dadas por el World Wide Web consortium W3C) y otras organizaciones internacionales acerca de cómo crear e interpretar documentos basados en la web. Su objetivo es crear una web que trabaje mejor para todos, con sitios accesibles a más personas y que funcionen en cualquier dispositivo de acceso a Internet.

Se puede comprobar de manera online si un documento Web cumple el estándar definido por W3C mediante https://validator.w3.org/

Actualmente la mayoría de navegadores aceptan páginas no estándar, pero cuanto más estándar se la aplicación web desarrollada, mayor probabilidad que funcione correctamente en todos los navegadores.

Es una práctica imprescindible el comprobar que cualquier desarrollo Web funcione correctamente en los principales navegadores.

2.3.- Principales navegadores.



La comunicación entre el servidor web y el navegador se realiza mediante el protocolo HTTP,

- **Microsoft Edge (Antiguo Internet Explorer)**
  - URL Oficial: https://www.microsoft.com/es-es/windows/microsoft-edge.
  - Antiguamente se llamaba Internet Explorer. Microsoft Edge está diseñado para ser un navegador web ligero con un motor de renderizado de código abierto construido

en torno a los estándares web.

- Utiliza una interfaz minimalista e integra plataformas online de Microsoft; el asistente Cortana para control de voz, funcionalidad de búsqueda e información dinámica relacionada con los sitios web en la barra de direcciones, …
- Resulta más rápido que su competencia (Chrome y Mozilla) debido al uso del motor Chakra como intérprete de Jscript (versión de Javascript propia de Microsoft)

- **Mozilla Firefox**
  - URL Oficial: https://www.mozilla.org/es-ES/firefox/new/
  - Mozilla Firefox es un navegador web libre y de código abierto desarrollado por la Corporación Mozilla y la Fundación Mozilla. Usa el motor Gecko para renderizar páginas webs, el cual implementa actuales y futuros estándares web.
  - Posee gran variedad de utilidades, extensiones y herramientas para la personalización y apariencia del navegador.
    - Posee una versión para desarrolladores: “Firefox Developer Edition” https://www.mozilla.org/en-US/firefox/developer/?utm_source=firebug&utm_medium=lp&utm_campaign=switch&utm_content=landingpage
- **Google Chrome**
  - URL Oficial: https://www.google.com/chrome/
  - Google Chrome es un navegador web desarrollado por Google y compilado con base en varios componentes e infraestructuras de desarrollo de aplicaciones (frameworks) de código abierto, como el motor de renderizado Blink (bifurcación o fork de WebKit). Está disponible gratuitamente bajo condiciones específicas del software privativo o cerrado.
  - Sus características son la seguridad, velocidad y estabilidad. Su rapidez y seguridad en los test comparativos se debe a que sigue una arquitectura multiproceso en la que cada pestaña se ejecuta de forma independiente.
- **Safari**
  - URL oficial: http://www.apple.com/es/safari/
  - Safari es un navegador web de código cerrado desarrollado por Apple Inc. Está disponible para OS X, iOS (el sistema usado por el iPhone, el iPod touch y iPad) y Windows (sin soporte desde el 2012).
  - Las últimas versiones incorporan la navegación por pestañas, corrector ortográfico en formularios, almacenamiento de direcciones favoritas, bloqueador de ventanas emergentes, soporte para motores de búsqueda personalizado o un gestor de descargas propio.

- **Opera**
  - URL oficial: http://www.opera.com/es
  - Opera es un navegador web creado por la empresa noruega Opera Software. Usa el motor de renderizado Blink. Tiene versiones para escritorio, teléfonos móviles y tabletas.

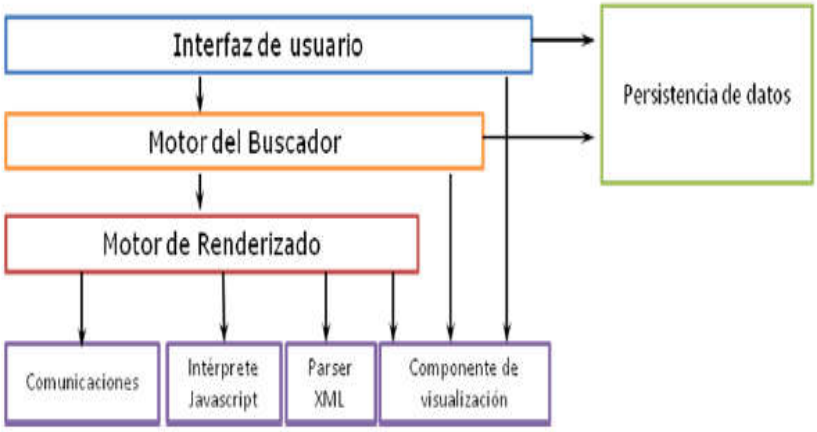
Los criterios que podemos seguir para diferenciar los navegadores son:

- Plataforma de Ejecución.
  - No todos los navegadores se pueden usar en cualquier ordenador.
  - Safari es exclusivo de Apple, pero tiene versiones para Windows.
- Características del navegador.
  - La mayoría añaden funcionalidades asociadas a la experiencia del usuario: administración de marcadores, gestores de descarga, almacenamiento seguro de contraseñas y datos de formulario, corrección ortográfica o definición de herramientas de búsqueda.
- Personalización de la interfaz.
  - Soporte para la navegación por pestañas, bloqueadores de ventanas emergentes, integración con visualizadores de formatos de ficheros (PDF), opciones de zoom o funciones avanzadas de búsqueda de texto.
- Soporte de tecnologías Web.
  - Nivel de soporte de tecnologías CSS, Java, lenguajes de scripting del cliente, RSS o Atom, XHTML, etc.
- Licencia de software.
  - De código libre, como Mozilla(licencia GNU GPL) y Google Chrome(licencia BSD).
  - Propietarios, como Internet Explorer o Safari.
  - Salvo raras excepciones (OmniWeb) todos son gratuitos.

3.- ARQUITECTURA DE EJECUCIÓN.

Cada navegador tiene su propia forma de interpretar la interacción con el usuario. El resultado de esta interacción, en cualquier caso, se inicia con el usuario indicado la dirección del recurso al que quiere acceder y termina con la visualización del recurso por parte del navegador en la pantalla. La forma de realizar este proceso depende del propósito del navegador y de la configuración del mismo.

Para poder llevar a cabo este proceso, cada navegador está formado por una serie de elementos y componentes que conforman lo que se denomina ***arquitectura del navegador***. A pesar de que cada navegador tiene su propia arquitectura, la mayoría coinciden en una serie de componentes básicos, es lo que se llama ***arquitectura de referencia***. Esta es:



Los componentes de esta arquitectura de referencia son:

- **Subsistema de interfaz de usuario**: es la capa que actúa de interfaz entre el usuario y el motor del buscador o de navegación. Ofrece funcionalidades como la virtualización de barras de herramientas, progreso de carga de página, gestión inteligente de las descargas, preferencias de configuración de usuarios impresión. Y en algunos casos puede comunicarse con el SO para el manejo de sesiones de usuario o el almacenamiento de preferencias de visualización y configuración.
- **Subsistema del motor del buscador o motor de navegación**: es un componente que ofrece una interfaz de alto nivel para el motor de renderizacdo. Su función principal es a de cargar una dirección determinada (URL) y soportar mecanismos básicos de navegación. Además, es el componente que gestiona las alertas de JavaScript y el proceso de carga de una agina. También es el encargado de consultar y administrar las preferencias del motor de renderización.
- **Subsistema de renderizado**: es el encargado de producir una presentación visual del recurso obtenido a partir del acceso a una dirección Web. El código de una página Web es interpretado por este módulo. En función de los lenguajes, estándares y tecnologías soportadas por el navegador, este módulo será capaz de mostrar documentos HTML o XML, hojas de estilo CSS, imágenes e incluso contenido embebido. Además, este

módulo Establece las dimensiones exactas de cada elemento a mostrar y la posición de estos.

- **Subsistema de Comunicaciones**: es el encargado de implementar los protocolos de transferencia de ficheros y documentos utilizados en Internet (HTTP, FTP, …). Además, es el responsable de identificar la codificación de los datos obtenidos en función de su tipo (texto, audio, video, …) codificado en estándar MIME (Multipurpose Intenet mail Extensions). Dependiendo de las capacidades personalizadas para este navegador, este subsistema puede almacenar una caché de elementos accedidos recientemente.
- **Intérprete de JaaScript**: las paginas HTML habitualmente llevan código intercalado para la provisión de ciertas funcionalidades al usuario como puede ser la respuesta a ciertos eventos del rato o del teclado. E lenguaje comúnmente aceptad para la programación de este código embebido es el JavaScript. El intérprete de JavaScript será el encargado de analizar y ejecutar el código. Este módulo puede ser configurado por cuestiones de seguridad o facilitar la navegación desde el motor de navegación o el motor de rendeizado. La existencia de módulos de interpretación de código libre de un navegador a otro. Para ello, es posible que existan subsistemas intérpretes de otros lenguajes, como applets de Jave , AJAX o ActionScript.
- **Parser XML** (analizador sintáctico): con el fin de poder acceder más fácilmente a los contenidos definidos en un documento HTML, los navegadores Web suelen incluir un módulo (paarser) que permite cargar en memoria una representación en árbol (árbol DOM Docuemnt Object Model) de la página. de esta forma, el acceso a los diferentes elementos de una página por parte del navegador es mucho más rápido.
- **Componente de virtualización**: ofrece funcionalidades relacionadas con la virtualización de los contenidos de un documento HTML en una página Web. Ofrece primitivas de dibujo y posicionamiento en una ventana, un conjunto de componentes visuales predefinidos (widgets) y un conjunto de fuente tipográficas. Suele estar relacionado con las librerías e visualización de Sistema Operativo.
- **Subsistema de persistencia de datos**: Funciona como almacén de diferentes tipos de datos para los principales subsistemas del navegador. Estos datos suelen estar relacionados con el almacenamiento de historiales de navegación y mantenimiento sede sesiones de usuario en disco. Otros datos de alto nivel que también son gestionados por este subsiste incluye las preferencias de configuración del navegador o el listado de marcadores. A bajo nivel, este sistema administra también los certificados de seguridad y las cookies.

4.- DISEÑO WEB.

El Diseño web es aquella actividad que consiste en planificar, diseñar, mantener y crear páginas web. Se aleja del término más tradicional del diseño en cuanto a que engloba una gran variedad de aspectos diferentes, como pueden ser el diseño de la interfaz, el diseño del material gráfico o incluso la experiencia del usuario con el sitio.

Es la principal disciplina a tener en cuenta a la hora de montar una web, ya que de ella depende el grado de usabilidad de la misma y la mejora de cara al visitante. Si este no recibe una experiencia agradable, con una interactividad, arquitectura de información, navegabilidad y usabilidad adecuadas, probablemente se desplace a la competencia o no saque real partido de lo que brinda la página.

Hay muchos más puntos que dependen del diseño web, como la tasa de conversión, el número de impresiones o incluso el posicionamiento orgánico en buscadores. Para el SEO, de hecho, el diseño es uno de los elementos más importantes, ya que puede determinar hasta qué punto una web es de calidad a ojos de motores de búsqueda como el de Google.

En la actualidad, es una de las disciplinas más demandadas y uno de los puntos en los que se hace mayor hincapié dentro de cualquier empresa. La presencia en Internet es fundamental, y el diseño es uno de los puntos más determinantes en este plano.

¿Para qué sirve el Diseño Web?

El Diseño web sirve para ofrecer a los usuarios una experiencia adecuada, suave y atractiva a la hora de moverse por una página en Internet. Es algo capaz de transmitir la imagen de una marca y su mensaje, a la vez que muestra el grado de compromiso de una firma por la buena experiencia de sus consumidores.

Un buen diseño, además, es capaz de responder bien a las necesidades del público actual, así como también marca la presencia de una firma en la red, ayudando en su posicionamiento para poder conseguir un mayor alcance y visibilidad.

4.1.- Áreas del Diseño Web.

Hay cinco áreas que cubren la mayor parte de las facetas del Diseño Web:

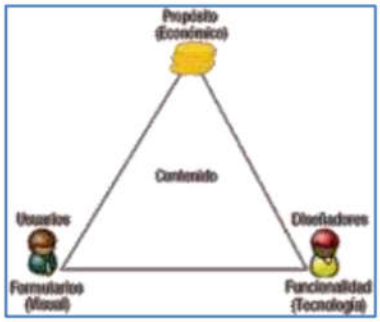
- **Contenido**: incluye la forma y organización del contenido del sitio. Esto puede abarcar desde cómo se escribe el texto hasta cómo está organizado, presentado y estructurado usando tecnologías de marcas como HTML.
- **Visual**: hace referencia a la plantilla empleada en un sitio web. Esta plantilla generalmente se genera usando HTML, CSS o incluso Flash y puede incluir elementos gráficos para decoración o para navegación. El aspecto visual es el aspecto más obvio del Diseño Web, pero no es la única disciplina o la más importante.
- **Tecnología**: aunque muchas de las tecnologías web como HTML o CSS entran dentro de esta categoría, la tecnología en este contexto generalmente hace referencia a los

diferentes tipos de elementos interactivos de un sitio web, generalmente aquellos construidos empleando técnicas de programación.

- **Distribución**: la velocidad y fiabilidad con la que un sitio web se distribuye en Internet o en una red interna corporativa está relacionado con el hardware/software utilizado y el tipo de arquitectura de red utilizada en la conexión.
- **Propósito**: la razón por la que un sitio web existe, generalmente está relacionada con algún aspecto de tipo económico. Por lo tanto, este elemento debería considerarse en todas las decisiones que tomemos en las diferentes áreas.

El porcentaje de influencia de cada una de estas áreas en un sitio web, puede variar dependiendo del tipo de sitio que se está construyendo. Una página personal generalmente no tiene las consideraciones económicas que tendría una web que va a vender productos en Internet.

Una forma de pensar en los componentes del Diseño Web es a través de la metáfora de la pirámide mostrada en la figura de la derecha. El contenido proporciona los ladrillos que formarán la pirámide, pero la base de la pirámide se fundamenta tanto en la parte visual como en la parte tecnológica y con el punto de vista económico puesto como objetivo o propósito final en la mayoría de los casos. Aunque la analogía de la pirámide es una forma un poco abstracta de describir el Diseño Web, es una herramienta que nos permite visualizar la interrelación de los diferentes componentes de la construcción Web.



Hoy en día los sitios web siguen un modelo basado en la **programación cliente-servidor** con tres **elementos** comunes:

- El lado del **servidor**: incluye el hardware y software del servidor Web, así como diferentes elementos de programación y tecnologías incrustadas. Las tecnologías pueden abarcar un rango amplio desde programas CGI escritos en PERL hasta aplicaciones multihilo basadas en Java, incluyendo tecnologías de servidor de bases de datos que soporten múltiples sitios web.
- El lado del **cliente**: hace referencia a los navegadores web y está soportado por tecnologías como HTML, CSS y lenguajes como JavaScript y controles ActiveX, los cuales se utilizan para crear la presentación de la página o proporcionar características interactivas.
- La **red**: describe los diferentes elementos de conectividad utilizados para mostrar el sitio web al usuario.

5.- LENGUAJES Y TECNOLOGÍAS DE PROGRAMACIÓN EN ENTORNO CLIENTE.

Cuando hablamos de tecnologías empleadas en lenguajes de programación web podemos citar dos grupos básicos: **cliente** y **servidor**. Las tecnologías cliente son aquellas que son ejecutadas en el cliente, generalmente en el contexto del navegador web. Cuando los programas o tecnologías son ejecutadas o interpretadas por el servidor estamos hablando de programación servidor.

Uno de los objetivos en la programación web es saber **escoger la tecnología correcta** para tu trabajo. Muchas veces los desarrolladores escogen rápidamente una tecnología favorita, que puede ser JavaScript, ColdFusion o PHP y la usan en todas las situaciones. La realidad es que cada tecnología tiene sus pros y sus contras. En general las tecnologías cliente y servidor poseen características que las hacen complementarias más que adversarias. Por ejemplo, cuando añadimos un formulario para recoger información y grabarla en una base de datos, es obvio que tendría más sentido chequear el formulario en el lado del cliente para asegurarnos que la información introducida es correcta, justo antes de enviar la información a la base de datos del servidor. La programación en el lado del cliente consigue que la validación del formulario sea mucho más efectiva y que el usuario se sienta menos frustrado al cubrir los datos en el formulario. Por otro lado, el almacenar los datos en el servidor estaría mucho mejor gestionado por una tecnología del lado del servidor, dando por supuesto que la base de datos estará en el lado del servidor.

Cada tipo general de programación tiene su propio lugar y la mezcla es generalmente la mejor solución. Cuando hablamos de lenguajes de programación en clientes web, podemos distinguir dos variantes:

- Lenguajes que nos permiten dar formato y estilo a una página web (HTML, CSS, etc.).
- Lenguajes que nos permite aportar dinamismo a páginas web (lenguajes de scripting).

Tabla comparativa de lenguajes de programación Web Cliente/Servidor	
Lado del Cliente	**Lado del Servidor**
— Aplicaciones de Ayuda. — Programas del API del navegador. • Plug-ins de Netscape. • Controles ActiveX. • Applets de Java. — Lenguajes de scripting. • JavaScript. • VBScript.	— Scripts y programas CGI. — Programas API del sevidor. • Módulos de Apache. • Extensiones ISAPI y filtros. • Servlets de Java. — Lenguajes de scripting. • PHP. • Active Server Pages (ASP/ASP.NET). • ColdFusion.

Para el desarrollo de este módulo hemos escogido JavaScript porque es el lenguaje de script (*Lenguaje de guiones o lenguaje de órdenes que se almacena por lo general en archivos de texto plano y que será ejecutado por un programa intérprete*) más utilizado en la programación en el lado del cliente, y está soportado la mayoritaria de las plataformas.

El desarrollo Web en el lado del cliente se da en 4 capas, estando en la capa superior el lenguaje JavaScript, gestionando el comportamiento de la página web.

Comportamiento (JavaScript).

Presentación (CSS)

Estructura (DOM/ Estructura HTML)

Contenido (texto, imágenes, videos, etc.)

} Contenido estructurado (documento HTML)

5.1.- Características.

Los lenguajes de programación para clientes web no son un reemplazo de la programación en el lado del servidor. Cualquier web que reaccione dinámicamente a interacciones del usuario o que almacene datos, estará gestionada por lenguajes de script en el lado del servidor, incluso aunque usemos JavaScript en el cliente para mejorar la experiencia de usuario. Las razones son simples:

- JavaScript por sí mismo no puede escribir ficheros en el servidor. Puede ayudar al usuario a elegir opciones o preparar datos para su envío, pero después de eso solamente podrá ceder los datos al lenguaje de servidor encargado de la actualización de datos.
- No todos los clientes web ejecutan JavaScript. Algunos lectores, dispositivos móviles, buscadores, o navegadores instalados en ciertos contextos están entre aquellos que no pueden realizar llamadas a JavaScript, o que simplemente son incompatibles con el código de JavaScript que reciben. Aunque esto ocurra, nuestra página web debería ser completamente funcional con JavaScript desactivado. Utilizaremos JavaScript para conseguir que la experiencia de navegación web sea lo más rápida, moderna o divertida posible, pero no dejaremos que nuestra web deje de funcionar si JavaScript no está funcionando.
- Uno de los caminos que más ha integrado la programación cliente con la programación servidor ha surgido gracias a **AJAX**. El proceso "asíncrono" de AJAX se ejecuta en el navegador del cliente y emplea JavaScript. Este proceso se encarga de solicitar datos XML, o enviar datos al lenguaje de servidor y todo ello de forma transparente en background. Los datos devueltos por el servidor pueden ser examinados por JavaScript en el lado del cliente, para actualizar secciones o partes de la página web. Es así como funciona una de las webs más populares en Internet, el servicio de Google Maps.

JavaScript está orientado a dar soluciones a:

- Conseguir que nuestra página web responda o reaccione directamente a la interacción del usuario con elementos de formulario y enlaces hipertexto.
- La distribución de pequeños grupos de datos y proporcionar una interfaz amigable para esos datos.

- Controlar múltiples ventanas o marcos de navegación, plug-ins, o applets Java basados en las elecciones que ha hecho el usuario en el documento HTML.
- Pre-procesar datos en el cliente antes de enviarlos al servidor.
- Modificar estilos y contenido en los navegadores de forma dinámica e instantáneamente, en respuesta a interacciones del usuario.
- Solicitar ficheros del servidor, y enviar solicitudes de lectura y escritura a los lenguajes de servidor.

Los lenguajes de script como JavaScript no se usan solamente en las páginas web. Los intérpretes de JavaScript están integrados en múltiples aplicaciones de uso cotidiano. Estas aplicaciones proporcionan su propio modelo de acceso y gestión de los módulos que componen la aplicación y para ello comparten el lenguaje JavaScript en cada aplicación. Podríamos citar varios ejemplos como: Google Desktop Gadgets, Adobe Acrobat, Dreamweaver, OpenOffice.org, Google Docs, etc.

5.2.- Compatibilidades.

A diferencia de otros tipos de scripts, JavaScript es interpretado por el cliente. Actualmente existen múltiples clientes o navegadores que soportan JavaScript, incluyendo Firefox, Google Chrome, Safari, Opera, Internet Explorer, etc. Por lo tanto, cuando escribimos un script en nuestra página web, tenemos que estar seguros de que será interpretado por diferentes navegadores y que aporte la misma funcionalidad y características en cada uno de ellos. Ésta es otra de las diferencias con los scripts de servidor en los que nosotros dispondremos del control total sobre su interpretación.

Cada tipo de navegador da soporte a diferentes características del JavaScript y además también añaden sus propios **bugs o fallos**. Algunos de estos fallos son específicos de la plataforma sobre la que se ejecuta ese navegador, mientras que otros son específicos del propio navegador en sí.

A veces las incompatibilidades entre navegadores al interpretar el código de JavaScript no vienen dadas por el propio código en sí, sino que su origen proviene del código fuente HTML. Por lo tanto, es muy importante que tu código HTML siga las **especificaciones del estándar W3C** y para ello dispones de herramientas como el validador HTML W3C:

También hay que tener precaución con las **limitaciones en el uso de JavaScript**:

- No todos los navegadores soportan lenguajes de script (en especial JavaScript) en el lado del cliente.
- Algunos dispositivos móviles tampoco podrán ejecutar JavaScript.
- Incluso las implementaciones más importantes de JavaScript en los diferentes navegadores no son totalmente compatibles entre ellas: por ejemplo, diferentes incompatibilidades entre Firefox e Internet Explorer.

- La ejecución de código JavaScript en el cliente podría ser desactivada por el usuario de forma manual, con lo que no podremos tener una confianza ciega en que se vaya a ejecutar siempre tu código de JavaScript.
- Algunos navegadores de voz, no interpretan el código de JavaScript.

5.3.- Seguridad.

JavaScript proporciona un gran potencial para diseñadores maliciosos que quieran distribuir sus scripts a través de la web. Para evitar esto, los navegadores web en el cliente aplican dos tipos de restricciones:

- Por razones de seguridad cuando se ejecuta código de JavaScript éste lo hace en un "espacio seguro de ejecución" en el cuál solamente podrá realizar tareas relacionadas con la web, nada de tareas genéricas de programación como creación de ficheros, etc.
- Además, los scripts están restringidos por la política de "mismo origen": la cual quiere decir que los scripts de una web no tendrán acceso a información tal como usuarios, contraseñas, o cookies enviadas desde otra web. La mayor parte de los agujeros de seguridad son infracciones tanto de la **política** de "**mismo origen**" como de la política de "espacio seguro de ejecución".

Al mismo tiempo es importante entender las **limitaciones que tiene JavaScript** y que, en parte, refuerzan sus capacidades de seguridad. JavaScript no podrá realizar ninguna de las siguientes tareas:

- Modificar o acceder a las preferencias del navegador del cliente, las características de apariencia de la ventana principal de navegación, las capacidades de impresión, o a los botones de acciones del navegador.
- Lanzar la ejecución de una aplicación en el ordenador del cliente.
- Leer o escribir ficheros o directorios en el ordenador del cliente (con la excepción de las cookies).
- Escribir directamente ficheros en el servidor.
- Capturar los datos procedentes de una transmisión en streaming de un servidor, para su retransmisión.
- Enviar e-mails a nosotros mismos de forma invisible sobre los visitantes a nuestra página web (aunque sí que podría enviar datos a una aplicación en el lado del servidor capaz de enviar correos).
- Interactuar directamente con los lenguajes de servidor.
- Las páginas web almacenadas en diferentes dominios no pueden ser accesibles por JavaScript.
- JavaScript es incapaz de proteger el origen de las imágenes de nuestra página.

- Implementar multiprocesamiento o multitarea.
- Otro tipo de **vulnerabilidades** que podemos encontrar están relacionadas con el XSS. Este tipo de vulnerabilidad viola la política de "mismo origen" y ocurre cuando un atacante es capaz de inyectar código malicioso en la página web presentada a su víctima. Este código malicioso puede provenir de la base de datos a la cual está accediendo esa víctima. Generalmente este tipo de errores se deben a fallos de implementación de los programadores de navegadores web.

Otro aspecto muy relacionado con la seguridad son los defectos o imperfecciones de los navegadores web o plugins utilizados. Estas imperfecciones pueden ser empleadas por los atacantes para escribir scripts maliciosos que se puedan ejecutar en el sistema operativo del usuario.

El **motor de ejecución de JavaScript** es el encargado de ejecutar el código de JavaScript en el navegador y por lo tanto es en él dónde recaerá el peso fuerte de la implementación de la seguridad. Podríamos citar varios ejemplos de motores de JavaScript:

- Active Script de Microsoft: tecnología que soporta JScript como lenguaje de scripting. A menudo se considera compatible con JavaScript, pero Microsoft emplea múltiples características que no siguen los estándares ECMA.
- El kit de herramientas Qt C++ también incluye un módulo intérprete de JavaScript.
- El lenguaje de programación Java en su versión JDK 1.6 introduce un paquete denominada javax. script que permite la ejecución de JavaScript.
- Y por supuesto todos los motores implementados por los navegadores web como Mozilla, Google, Opera, Safari, etc. Cada uno de ellos da soporte a alguna de las diferentes versiones de JavaScript.

Hoy en día una de las características que más se resalta y que permite diferenciar a unos navegadores de otros, es la rapidez con la que sus motores de JavaScript pueden ejecutar las aplicaciones, y la seguridad y aislamiento que ofrecen en la ejecución de las aplicaciones en diferentes ventanas o pestañas de navegación.

6.- INTEGRACIÓN DE CÓDIGO JavaScript CON HTML.

Para poder 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). Aquí vamos a ver 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.

Los navegadores web te permiten varias opciones de inserción de código de JavaScript. Podremos insertar código usando las etiquetas `<script> </script>` y empleando un atributo `type` indicaremos qué tipo de lenguaje de script estamos utilizando:

Por ejemplo:

```
<script type="text/javascript">
   // El código de JavaScript vendrá aquí.
</script>
```

Otra forma de integrar el código de JavaScript es incrustar un fichero externo que contenga el código de JavaScript. Ésta sería la forma más recomendable, ya que así se consigue una separación entre el código y la estructura de la página web y como ventajas adicionales podrás compartir código entre diferentes páginas, centralizar el código para la depuración de errores, tendrás mayor claridad en tus desarrollos, más modularidad, seguridad del código y conseguirás que las páginas carguen más rápido.

La rapidez de carga de las páginas se consigue al tener el código de JavaScript en un fichero independiente, ya que si más de una página tiene que acceder a ese fichero lo cogerá automáticamente de la caché del navegador con lo que se acelerará la carga de la página.

Para ello tendremos que añadir a la etiqueta script el atributo `src`, con el nombre del fichero que contiene el código de JavaScript. Generalmente los ficheros que contienen texto de JavaScript tendrán la extensión `.js`.

Por ejemplo:

```
<script type="text/javascript" src="tucodigo.js"></script>
```

Si necesitas cargar más de un fichero `.js` repite la misma instrucción cambiando el nombre del fichero. Las etiquetas de `<script>` y `</script>` son obligatorias a la hora de incluir el fichero `.js`. **Atención**: no escribas ningún código de JavaScript entre esas etiquetas cuando uses el atributo `src`.

Para **referenciar el fichero origen .js** de JavaScript dependerá de la localización física de ese fichero. Por ejemplo, en la línea anterior el fichero `tucodigo.js` deberá estar en el mismo directorio que el fichero `.html`. Podrás enlazar fácilmente a otros ficheros de JavaScript localizados en directorios diferentes de tu servidor o de tu dominio. Si quieres hacer una referencia absoluta al fichero, la ruta tendrá que comenzar por `http://`, en otro caso tendrás que poner la ruta relativa dónde se encuentra tu fichero `.js`.

Ejemplos:

```
<script type="text/javascript"
src="http://www.tudominio.com/ejemplo.js"></script>
<script type="text/javascript" src="../js/ejemplo.js"></script>
```

(el fichero `ejemplo.js` se encuentra en el directorio anterior (`../`) al actual, dentro de la carpeta j`s/`)

Cuando alguien examine el código fuente de tu página web verá el enlace a tu fichero `.js`, en lugar de ver el código de JavaScript directamente. Esto no quiere decir que tu código no sea inaccesible, ya que simplemente copiando la ruta de tu fichero `.js` y tecleándola en el

navegador podremos descargar el fichero `.js` y ver todo el código de JavaScript. En otras palabras, nada de lo que tu navegador descargue para mostrar la página web podrá estar oculto de la vista de cualquier programador.

A veces te puedes encontrar que tu script se va a ejecutar en un navegador que no soporta JavaScript. Para ello dispones de una etiqueta **<noscript>**Texto informativo **</noscript>** que te permitirá indicar un texto adicional que se mostrará indicando que ese navegador no soporta JavaScript.

Si estamos trabajando con **XHTML**, la sintaxis para insertar un código de **JavaScript** directamente en el XHTML es un poco diferente:

```
<script type="text/javascript">
   <!--//--><![CDATA[//><!--
      // código de JavaScript a continuación
   //--><!]]>
</script>
```

Los analizadores de XHTML son intolerantes a los elementos que no comienzan por **<** y a las entidades HTML que no comienzan por **&,** y estos caracteres como verás más adelante son ampliamente utilizados en JavaScript. La solución es encapsular las instrucciones de JavaScript en lo que se conoce como sección **CDATA**:

```
<![CDATA[
   XHTML permite cualquier tipo de carácter aquí incluyendo < y el símbolo &
]]>
```

Como puedes observar el código es un poco complejo en este caso, razón de más para integrar el código de JavaScript en un fichero externo.