1 Selección de arquitecturas y herramientas de programación

OBJETIVOS DEL CAPÍTULO

- ✓ 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

En este primer capítulo presentamos los conceptos necesarios para comprender el contexto de ejecución de páginas web como forma de acceder a recursos de aplicaciones y sistemas de información web. Para ello, introducimos las características principales de los navegadores más comunes utilizados hoy en día. Describiremos, así mismo, los diferentes lenguajes y tecnologías de programación, del lado del cliente, aplicables en este tipo de entornos. Por último, se hace un recorrido introductorio por algunas de las técnicas de integración que el desarrollador tiene a su alcance para intercalar código con las etiquetas HTML.

1.1 EVOLUCIÓN Y CARACTERÍSTICAS DE LOS NAVEGADORES WEB

La *World Wide Web* (o "la Web", como se conoce comúnmente) representa un universo de información accesible globalmente a través de la red Internet. Está formada por un conjunto de recursos interconectados que conforman el conocimiento humano actual. El funcionamiento 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 (*hubs*, repetidores, puentes, pasarelas, encaminadores, etc.) y los protocolos de comunicaciones (TCP, IP, HTTP, FTP, SMTP, etc.), hasta la utilización del sistema de nombres de dominio (DNS) para la búsqueda y recuperación de recursos o la utilización de software específico para proveer y consumir dichos recursos.

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 está robustamente cimentada en el intercambio de mensajes como el único elemento de acoplamiento entre ambos. En este libro, y en este capítulo en concreto, nos vamos a centrar en las características de los componentes software que se utilizan en el cliente.

Uno de los componentes más habituales en el cliente es el navegador web, que permite acceder al contenido ofrecido por los servidores de Internet sin la necesidad de que el usuario instale un nuevo programa (con excepciones). Podemos encontrarnos muchos tipos de clientes en función de sus capacidades, los lenguajes soportados o las facilidades de configuración. Los más livianos, o "ligeros", son los que por sí solos no pueden ejecutar ninguna operación real más allá de la de conectarse al servidor. Sin embargo, actualmente la tendencia es a disponer de clientes complejos, que utilizan lenguajes como Java o funciones avanzadas en DHTML para otorgar mayor funcionalidad y flexibilidad al usuario. Estos navegadores pueden no solo conectarse al servidor, sino que también son capaces de procesar o sincronizar datos para su uso sin necesidad de que el usuario intervenga.

En cualquier caso, debemos entender que un navegador web, o explorador web (*browser*), es una aplicación, distribuida habitualmente como software libre, que permite a un usuario acceder (y, normalmente, visualizar) a un recurso publicado por un servidor web a través de Internet y descrito mediante una dirección URL (*Universal Resource Locator*). Como ya hemos dicho, lo más habitual es que utilicemos los exploradores web para "navegar" por recursos de tipo hipertexto, comúnmente descritos en HTML, ofrecidos por servidores web de todo el mundo a través de Internet.

Desde la creación de la Web a principios de los años 90, los navegadores web han evolucionado desde meros visualizadores de texto que, aunque no ofrecían capacidades multimedia (visualización de imágenes), cumplían su propósito (Links, Lynx, W3M); hasta los actuales navegadores, totalmente preparados para soportar cualquier tipo de

interacción y funcionalidad requerida por el usuario. A continuación describimos una pequeña lista de algunos de los exploradores más relevantes a lo largo de la corta historia de los clientes de navegación web:

- **Mosaic**. Se considera uno de los primeros navegadores web y 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 abandonó en 1997.

- **Netscape Navigator** (después **Communicator**). Fue el primer navegador en incluir un módulo para la ejecución de código *script* (JavaScript). Se le considera como 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**. Es el 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 debido al aumento de usuarios que optan por otros navegadores, como Firefox o Chrome. A fecha de finales de 2011, la última versión publicada es la 9.0, que incorpora numerosos avances en cuanto a soporte de 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 de la década de los 2000 venía teniendo Internet Explorer.

- **Google Chrome**. De reciente creación (septiembre de 2008), es el navegador de Google compilado a partir de componentes de código abierto. En boca de sus desarrolladores, sus características principales son 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 arquitectura multiproceso en la que cada pestaña es ejecutada de forma independiente.

- **Safari**. Es el navegador por defecto de los sistemas de Apple, aunque también se han desarrollado versiones para su 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 o un gestor de descargas propio.

- **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. Utiliza un motor de renderizado de páginas similar al de Chrome o Safari.

En la siguiente gráfica puede verse una tendencia de la utilización de navegadores en el período 2008-2012 (obtenida de *gs.StatCounter.com*):

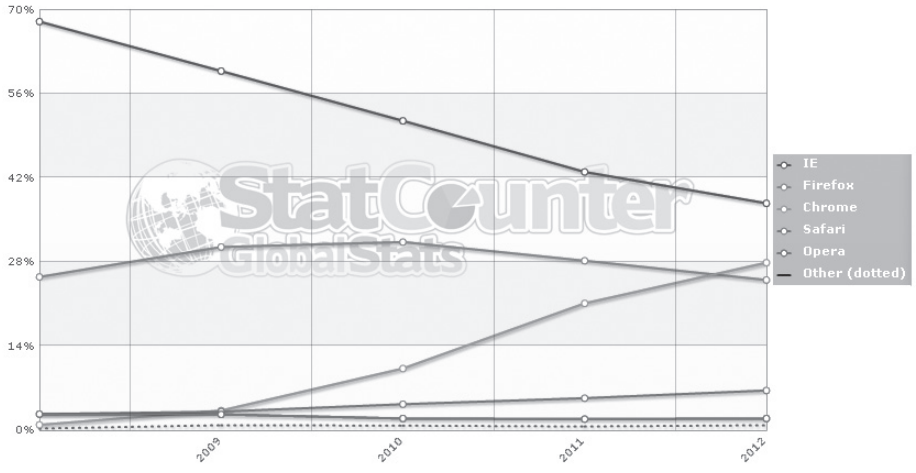


Figura 1.1. *Estadísticas de uso de navegadores (2008-2012)*

Podemos diferenciar los navegadores anteriores de acuerdo a una serie de criterios:

- **Plataforma de ejecución**. No todos los navegadores pueden ser ejecutados en cualquier sistema operativo. Safari, por ejemplo, es exclusivo de los sistemas de Apple aunque tiene versiones para Windows.
- **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. Algunas de las características soportadas de forma nativa (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.
- **Personalización de la interfaz**. Las funciones de accesibilidad que definen la experiencia del usuario con un navegador web también son un aspecto diferencial. Entre los aspectos más destacados podemos mencionar el soporte para la navegación por pestañas, la existencia de 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.
- **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 o Atom (sindicación de contenidos), XHTML (HTML con formato de XML), etc.
- **Licencia de software**. Existen navegadores de código libre, como Mozilla Firefox (licencia GNU GPL) o Google Chrome (licencia BSD), y navegadores propietarios como Internet Explorer (Microsoft) o Safari (Apple). Salvo raras excepciones (OmniWeb) todos los navegadores son gratuitos.

1.2 ARQUITECTURA DE EJECUCIÓN

Cada navegador web tiene su propia forma de interpretar la interacción con un 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 *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 que pueden verse en la siguiente figura:

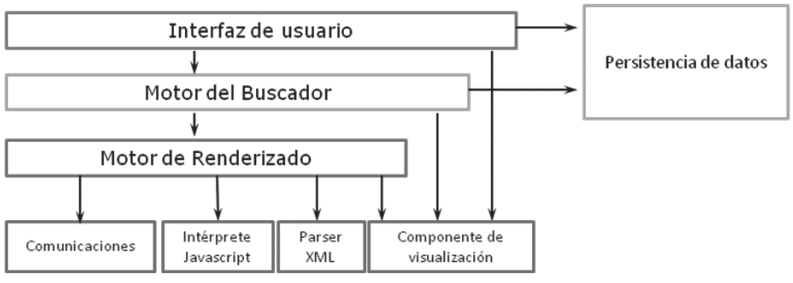


Figura 1.2. *Arquitectura de referencia de un navegador web*

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 tales como la visualización de barras de herramientas, progreso de carga de la página, gestión inteligente de las descargas, preferencias de configuración de usuario o 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.
- **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. Su función principal es la de cargar una dirección determinada (URL o URI) y soportar los mecanismos básicos de navegación tales como ir a la página anterior o siguiente o la recarga de la página. Además, es el componente que gestiona las alertas de JavaScript (mensajes de usuario) y el proceso de carga de una página (es quien le provee de información a la interfaz de usuario al respecto). Finalmente, es el encargado de consultar y administrar las preferencias de ejecución del motor de renderizado.

- **Subsistema de renderizado**. Este componente es el encargado de producir una representación 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, XML, hojas de estilo CSS e incluso contenido embebido en la página (audio/vídeo) e imágenes. Además, este módulo establece las dimensiones exactas de cada elemento a mostrar y, en ocasiones, es el responsable de 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.
  - *WebKit*, el motor de Google Chrome, Epiphany y Safari.
  - *Presto*, el motor de Opera.
  - *Tasman*, el motor de Internet Explorer para Mac.

- **Subsistema de comunicaciones**. Es el subsistema encargado de implementar los protocolos de transferencia de ficheros y documentos utilizados en Internet (HTTP, FTP, etc.). Además, es el responsable de 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*). Dependiendo de las capacidades personalizadas para el navegador, este subsistema puede almacenar una caché de elementos accedidos recientemente.

- **Intérprete de JavaScript**. Las páginas HTML habitualmente llevan código intercalado para la provisión de ciertas funcionalidades al usuario como puede ser la respuesta a ciertos eventos del ratón o del teclado. El lenguaje comúnmente aceptado para la programación de este código embebido es JavaScript (siguiendo el estándar ECMAScript). El intérprete de JavaScript será el encargado de analizar y ejecutar dicho código. 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.

- **Parser XML**. Con el fin de poder acceder más fácilmente a los contenidos definidos en un documento HTML (en realidad XHTML), los navegadores web suelen incluir un módulo (*parser*) que permite cargar en memoria una representación en árbol (árbol DOM, *Document 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 visualización**. Este subsistema ofrece funcionalidades relacionadas con la visualización 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 fuentes tipográficas a los subsistemas principales del navegador web. Suele estar muy relacionado con las librerías de visualización del 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 el mantenimiento de sesiones de usuario en disco. Otros datos de alto nivel que también son gestionados por este subsistema incluyen las preferencias de configuración del navegador (de barras de herramientas, por ejemplo) o el listado de marcadores. A bajo nivel, este sistema administra también los certificados de seguridad y las *cookies*.

1.3 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 arquitectura Cliente/Servidor. El lenguaje cliente principal es HTML (lenguaje de marcado de hipertexto, *HyperText Markup Language*), ya que la mayoría de páginas del 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*) o XHTML (*eXtensible HTML*). Con el fin de mejorar la interactividad con el usuario, en este grupo de lenguajes cliente podemos incluir todos los lenguajes de *script*, tales como JavaScript (el más utilizado dentro de esta rama) o VBScript. También existen otros lenguajes más independientes, como ActionScript (para crear contenido Flash) o AJAX (como extensión a JavaScript para comunicación asíncrona). A continuación vamos a analizar brevemente estos lenguajes.

1.3.1 HTML Y DERIVADOS

HTML es una particularización de un lenguaje anterior, SGML (*Standard Generalized Markup Language*), un sistema para 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 *Hyper Text 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; y 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 por 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 de documento. El entorno para desarrollar HTML puede ser simplemente un procesador de textos.

Con el lenguaje HTML se pueden hacer gran variedad de acciones, desde organizar simplemente el texto y los objetos de una página web, pasando por crear listas y tablas, hasta llegar a 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 este, 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.
- El lenguaje **XHTML** es también muy similar al lenguaje HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje XML. Técnicamente, como ya hemos comentado, HTML es descendiente directo del lenguaje SGML, mientras que XHTML lo es del XML (que, a su vez, también es descendiente 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.
- El HTML Dinámico (**DHTML**) 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 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 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 poder utilizarlo.

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.

1.3.2 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 queramos 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.

1.3.3 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 consideran "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 en capítulos posteriores.

1.3.4 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 los 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 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 las páginas web y que el navegador ejecuta en el espacio de la página (a través de un módulo o extensión concretos).

Los *applets* se programan en Java y se envían al cliente precompilados, es por ello que la manera de trabajar de estos varía un poco con respecto a los lenguajes de *script* como JavaScript. Una de las principales ventajas de utilizar *applets* es que son mucho menos dependientes del navegador que los *scripts* en JavaScript, incluso independientes del sistema operativo del ordenador donde se ejecutan. Además, en principio Java es más potente que JavaScript, por lo que el número de aplicaciones de los *applets* podrá ser mayor.

Como desventajas notables frente al uso de JavaScript, cabe señalar que los *applets* son más lentos de procesar y que 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. Como consecuencia directa y, en general, con los *applets* de Java no podremos realizar directamente acciones tales como abrir ventanas secundarias, controlar marcos (*frames*), obtener la información de formularios, administrar capas, etc.

1.3.5 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 aplicaciones web interactivas. El primer aspecto que define a AJAX es que este tipo de aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios que acceden a una página web. La segunda característica es que, al contrario que con una página web HTML/XHTML/DHTML, en la que la comunicación se interrumpe una vez el cliente recibe la página, con AJAX se mantiene una comunicación asíncrona con el servidor en segundo plano (sin que el usuario sea consciente de dicha comunicación). La consecuencia directa de esta técnica es que podemos realizar cambios sobre las páginas del cliente sin necesidad de que éste proceda a recargarlas. Este hecho implica un aumento de la interactividad con el usuario y de la 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:

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

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. Entre las desventajas que pueden detectarse con respecto al desarrollo de aplicaciones cliente con AJAX podemos mencionar una curva de aprendizaje del desarrollo de este tipo de aplicaciones más elevada o el hecho de que, al tratarse de comunicaciones asíncronas, el navegador no es capaz de guardar un historial real de los contenidos a los que se ha accedido.

1.3.6 ADOBE FLASH Y ACTIONSCRIPT

Flash es una tecnología de animación actualmente bajo licencia de Adobe 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 dibujo y animación 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 HMTL 5.

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 tecnologías 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 o que se trate de un software 100% propietario, coartando la libertad de extender o mejorar el núcleo de Flash.

1.4 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 esta sección 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. Las particularidades de otras tecnologías se verán en los capítulos correspondientes.

La integración de JavaScript y HTML/XHTML podemos hacerla de diferentes formas. Esta flexibilidad se refleja en, al menos, tres formas de incluir código JavaScript en páginas web.

1.4.1 JAVASCRIPT EN EL MISMO DOCUMENTO HTML

HTML se basa en el uso de unas etiquetas predefinidas para marcar el texto. Una de estas etiquetas es `<script>` (y `</script>` para indicar la finalización de un bloque de 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 contenido 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/DTD/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.

1.4.2 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/DTD/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 2</title>
    <script type="text/javascript"
      src="/inc/mensaje.js"></script>
  </head>
  <body>
    <h1>Ejemplo 2: fichero externo</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 una carpeta denominada `inc`. 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 de 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.

1.4.3 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/DTD/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 3</title>
  </head>
  <body>
    <p onclick="alert('Prueba de JavaScript');">
      Ejemplo 3: código 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.

ACTIVIDADES 1.1

- En este capítulo se ha mencionado la "Guerra de los Navegadores". Se propone que el alumno profundice en este período, identifique sus consecuencias y debata con los compañeros acerca del momento actual (considerado como la "Segunda Guerra de los Navegadores").
- La arquitectura de referencia presentada en la Figura 1.2 puede aplicarse a cualquier navegador. Se propone que el alumno aplique esta arquitectura investigando los diferentes módulos y componentes de un navegador actual como Google Chrome.
- La tecnología Flash está siendo sustituida progresivamente por capacidades similares dentro del estándar HTML 5. Se propone que el alumno investigue cuál es el soporte que HTML 5 propone para la visualización de contenido multimedia (puede consultarse en la página del estándar HTML del W3C).
- La forma de vincular un fichero externo mediante el atributo `src` de la etiqueta `<script>` requiere una ruta relativa. Se propone que el alumno estudie las reglas de esta vinculación y los metacaracteres permitidos (puede consultarse en la página del estándar HTML del W3C).

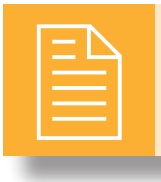
RESUMEN DEL CAPÍTULO

El desarrollo de aplicaciones en el lado del cliente parte de la base de un componente software específico que permite acceder a los contenidos ofrecidos por un servidor, como parte de la arquitectura Cliente/Servidor, a través de Internet. Este componente se conoce con el nombre de "navegador web". En los últimos años muchos tipos de navegadores web han aparecido con diferentes características y para ser ejecutados en diferentes plataformas. Un aspecto común a todos ellos es el soporte para la gran mayoría de tecnologías que se pueden utilizar en el desarrollo de aplicaciones en el entorno del cliente. Entre estas tecnologías destaca HTML como lenguaje de descripción de marcas en documentos de texto accedidos a través de la Web. Otros lenguajes y tecnologías, como JavaScript, CSS, AJAX, Flash o los *applets* de Java, han permitido dotar al cliente de numerosas herramientas que facilitan la interactividad con el usuario mejorando su experiencia de navegación en la Web.

Finalmente, debemos ser conscientes de las restricciones y particularidades que cada lenguaje requiere si queremos integrar estas tecnologías con el lenguaje HTML y crear páginas web atractivas y dinámicas.

EJERCICIOS PROPUESTOS

1. Implementar los ejemplos propuestos en el apartado 1.4 y probarlos con los navegadores web más utilizados que aparecen en la Figura 1.1.



TEST DE CONOCIMIENTOS

1 Señale la respuesta correcta con respecto a las características de un navegador web:
- **a)** No pueden realizar peticiones al servidor sin intervención directa del usuario.
- **b)** Todos los navegadores web actuales son gratuitos.
- **c)** Todos los navegadores actuales soportan la ejecución de JavaScript.
- **d)** No pueden almacenar ningún tipo de dato relacionado con la navegación del cliente.

2 ¿Cuál de los siguientes módulos no forma parte de la arquitectura de referencia de un navegador web?
- **a)** Módulo de gestión de usuarios.
- **b)** Módulo de persistencia de datos.
- **c)** Módulo de comunicaciones.
- **d)** Motor de renderizado.

3 Señale la respuesta correcta con respecto a las características de XHTML:
- **a)** Al igual que HTML, no tiene por qué estar bien construido para que sea válido.
- **b)** Es la unión de los principios de XML con el lenguaje HTML.
- **c)** No permite la inclusión de código JavaScript.
- **d)** Ninguna los anteriores.

4 Señale la respuesta correcta con respecto a JavaScript:
- **a)** No está soportado por ninguno de los navegadores actuales.
- **b)** Es un estándar que indica la forma en la que un servidor debe ejecutar un programa externo.
- **c)** No se puede utilizar para programar *applets*.
- **d)** Se puede utilizar para programar Flash.

5 ¿Cuál de los siguientes fragmentos de código sería correcto para integrar código JavaScript en HTML?
- **a)** `<h1 onclick="<script>alert("Prueba de JavaScript "); </script>">`.
- **b)** `<script type="text" src="/inc/mensaje.js"> </script>`.
- **c)** `<h1 src="/inc/mensaje.js">Mensaje</h1>`.
- **d)** Ninguna de las anteriores.