*I've seen the* **FUTURE**

*It's in my* **BROWSER**

# HTML5: ¿nuevas perspectivas o déjà vu?

Imagen: CC-BY W3C (http://www.w3.org).

*Nelson Baloian*
*Profesor Asociado DCC, U. de Chile. Doktor rer. nat, Universität Duisburg, Alemania (1997); Ingeniero Civil en Computación, Universidad de Chile (1988). Líneas de especialización: Instrucción Asistida por Computador, Trabajo Colaborativo, Sistemas Distribuidos.*
*nbaloian@dcc.uchile.cl*

## LA NECESIDAD DE ESTÁNDARES EN LA COMPUTACIÓN

Los estándares son de gran ayuda en cualquier disciplina, pues permiten que distintas entidades puedan desarrollar diversos artefactos que pueden interactuar o complementarse sin necesidad que se comuniquen explícitamente, ya que el estándar cumple dicha función. En computación esto es tremendamente cierto, ya que tenemos millones de entidades de los más variados tamaños y formas de organización desarrollando software en todo el mundo. No pocas veces hay empresas que se han sentido con la fortaleza suficiente para lanzar al mercado productos que difieren de las normas, como lo hizo IBM con el sistema de codificación EBCDIC.

Es difícil imaginar que la computación sería lo que es hoy, sin estándares como TCP/IP, HTTP y HTML. Si bien hasta hace algunos años parecía que la computación se estabilizaba alrededor de tres plataformas principales (Windows, Unix/Linux y Mac) la irrupción masiva de la computación móvil ha sumado una serie de diversas plataformas al paisaje computacional, complicando así el panorama de los desarrolladores. No sólo hay diferencias grandes en el desarrollo de aplicaciones para equipos fijos y móviles, sino también entre los mismos equipos móviles hay variedades irreconciliables. Y como si esto fuera poco, tenemos una variedad de tamaños de pantalla, dificultando el diseño de una interfaz de aplicación que sea compatible con todos los equipos.

Es por todo esto que algunos desarrolladores han puesto sus ojos en HTML5 como una herramienta que les permite desarrollar aplicaciones altamente interactivas que sean capaces de correr en varias plataformas. Para ponerlo en pocas palabras, podríamos

decir que HTML5 se diferencia de su predecesor HTML4 en cuanto a que este último incorpora una serie de elementos nuevos que pueden ser instanciados y manipulados desde pedazos de código JavaScript insertos en la página web.

## NUEVOS ELEMENTOS PRESENTES EN HTML5

En efecto, HTML5 provee nuevas funcionalidades que permiten enriquecer significativamente sitios desarrollados con HTML a través de nuevos elementos que permiten usar nuevas técnicas de presentación, comunicación y manejo de datos. A continuación nombramos algunos de estos elementos:

**Canvas:** a través del objeto CANVAS HTML5 provee la posibilidad de incorporar gráficos 2D en forma fácil y flexible, además de capturar y procesar los eventos de interacción del usuario. Su función se parece mucho al canvas de Java, permitiendo dibujar líneas simples, elipses, rectángulos, etc. Además se pueden colocar imágenes sobre un CANVAS que pueden ser movidas, rotadas y escaladas (cambio de tamaño). También se pueden manejar algunas características gráficas como transparencia. Todas estas funciones están a disposición del desarrollador como simples comandos en JavaScript. Un ejemplo básico del uso de éste se ve en la Figura 1. Este código define un elemento CANVAS de tamaño 500x250 píxeles y de color verde, como se muestra en la Figura al hacer clic sobre él.

**WebSockets:** dado el protocolo HTTP, un sitio web sólo permite que el servidor reaccione a un requerimiento de un cliente del tipo request-replay, permitiendo un modo de comunicación esencialmente unidireccional. La única forma de que el cliente recibiera datos del servidor cuando el servidor los tuviera disponible era mediante métodos de "pull" (por ej., AJAX) que básicamente consisten en que el cliente pregunta a intervalos constantes si existe nueva información. Esto cambia con la implementación de WebSockets, ya que este enfoque provee una comunicación bidireccional, tal como en los sockets tradicionales de TCP/IP pero sobre protocolo HTTP. Para recibir datos desde un WebSocket, la página debe implementar los siguientes métodos en JavaScript, los cuales serán invocados por el navegador cuando sucedan los eventos correspondientes:

- **onopen:** se llama cuando se abre un WebSocket, es decir, cuando la comunicación se establece.
- **onmessage:** se llama cuando un mensaje desde el servidor es recibido, el mensaje es parámetro de este método.
- **onclose:** se llama cuando la conexión al servidor es cerrada.

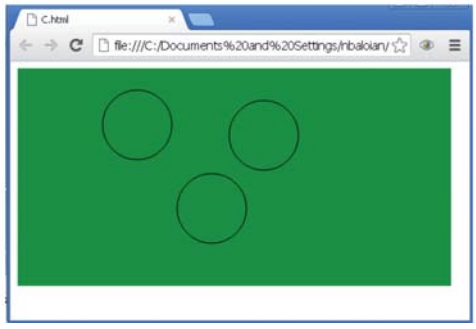
**Web Database:** HTML5 provee una implementación de SQLite que permite asociar aplicaciones Web con una base de datos local, permitiendo guardar datos recolectados en forma remota en una base de datos local.

**LocalStorage:** consiste en una tabla de hashing que permite guardar hasta 10Mb de pares (nombre, valor); estos datos están relacionados con la aplicación web en modo online y offline.

**FileSystem, Manifest y modo offline:** la clave para el desarrollo de aplicaciones web es el uso combinado del sistema de archivos FileSystem y el archivo de Manifest. FileSystem es una API que proporciona un sistema de archivos independiente de otras aplicaciones que estén corriendo y de los archivos de usuario. Manifest es una declaración de cuáles son los archivos que componen la aplicación proveyendo referencias locales a ellos. El Manifest, es utilizado por los navegadores para descargar estos archivos y cargarlos a un

**Figura 1**

```
<canvas style="top:0;left:0; background-color:#180" id="can" width="500" height="250" ></canvas>
<script>
// get the canvas element and its context
var canvas = document.getElementById('can');
var context = canvas.getContext('2d');
function draw(event){
   context.beginPath();
   context.arc(event.pageX,event.pageY,40,0,2*Math.PI);
   context.stroke();
}
// attach click event listener
canvas.addEventListener('click',draw, false);
</script>
```




**El código en HTML y JavaScript, y su resultado.**

sistema de archivos local, lo que permite el funcionamiento de la aplicación sin conexión.

**Geolocation:** dadas las capacidades de los dispositivos móviles actuales, hay muchas aplicaciones para ellos que utilizan la API para hacer uso de esta información. HTML5 también provee acceso a esta API, que permite conocer las coordenadas donde se encuentra el dispositivo en el momento, y así integrar esta información a la aplicación. Los datos disponibles son: latitud, longitud, altitud, exactitud de la medida de altitud provista y velocidad de desplazamiento.

**WebWorker:** permite al desarrollador trabajar con hilos de ejecución (threads) en un entorno web. Normalmente, el navegador es quien controla todos los threads activos corriendo dentro de su ambiente y estos no son referenciables desde el código JavaScript inserto en la página que se está interpretando. Sin embargo, un WebWorker permite operaciones síncronas y asíncronas, sobre threads independientemente de las que maneje el navegador.

**WebGL:** adicionalmente a las capacidades de dibujo 2D que provee el CANVAS de HTML5 también se provee el manejo de dibujos 3D basado en OpenGL, llamado WebGL. Al hacer uso de esta tecnología es posible enriquecer la presentación de las interfaces gráficas basadas en HTML5.

La lista de elementos adicionales que tiene HTML5 nos hace pensar que algunos de ellos fueron diseñados con la idea de apoyar (también) la computación móvil. En efecto, el uso de la geolocalización tiene mayor sentido cuando el usuario está en movimiento, o cuando la posición del usuario varía más o menos frecuentemente. También los elementos que permiten el almacenamiento local con conexión y sin conexión ayudan al desarrollo de este tipo de aplicaciones, pues en el escenario móvil es común que se pierda la conexión a ratos y, por lo tanto, conveniente que se trabaje con datos localmente hasta que se vuelva a tener conexión. Suponemos que esto no es casualidad, pues es justamente en el ámbito de la computación móvil donde un estándar se hace hoy en día urgentemente necesario, dada la heterogeneidad de los dispositivos disponibles en el mercado.

## DESAFÍOS EN LOS ESCENARIOS MÓVILES DE HOY

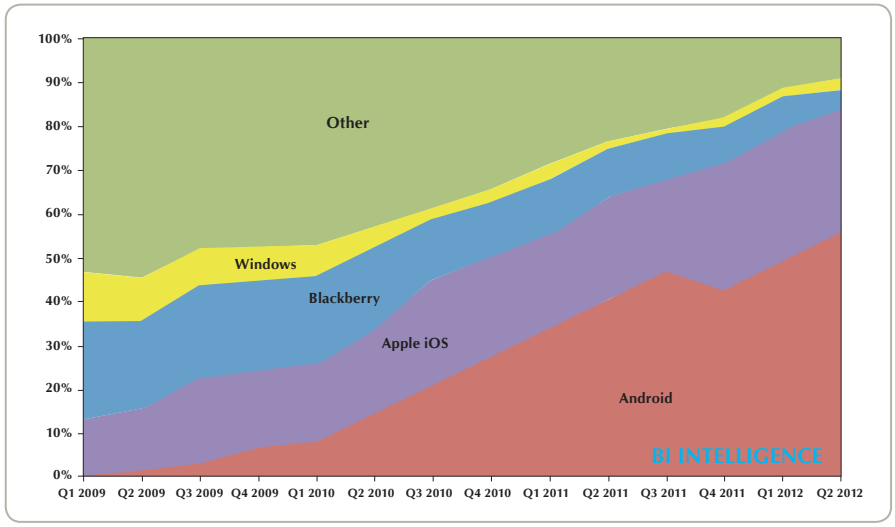
Hoy nos enfrentamos a varios desafíos al tratar de desarrollar aplicaciones móviles. Como describe G.Avellis et al: 2003, algunos de los desafíos podrían ser resueltos por un enfoque arquitectónico que permite la adaptación flexible de las diferentes representaciones de los mismos datos. Con este planteamiento varios problemas, como las pantallas de diferentes tamaños, diferentes mecanismos de interacción y el poder computacional, pueden ser abordados de manera satisfactoria.

Otro desafío importante desde el punto de vista técnico en escenarios móviles de hoy es la heterogeneidad de las plataformas de sistemas operativos móviles. Por lo menos hay cinco grandes jugadores actualmente en el mercado: iOS, Android, Symbian, Blackberry y WindowsMobile. Todos ellos proporcionan interesantes plataformas de desarrollo integrado que permiten el desarrollo de aplicaciones específicas para la plataforma.

De acuerdo con Álex Cocotas (http://www.businessinsider.com/mobile-platform-market-share-2012-8), la distribución de las ventas de dispositivos móviles por proveedor de los diferentes sistemas operativos móviles, en el tercer cuarto de 2012 fue la siguiente: el sistema operativo más extendido es el móvil Android (58%). Segundo está iOS (26%), sistema de los teléfonos móviles de Apple. Le sigue Blackberry con cerca del 5%, Windows Mobile apenas un 3% y Otros cerca del 11% (ver Figura 2). Si bien se podría pensar en que Android domina el mercado, se ha visto que éste es bastante dinámico y aún no se puede decir quién tendrá la última palabra.

Por lo tanto, deben considerarse alternativas distintas de la implementación de las aplicaciones dependientes de la plataforma para dispositivos móviles. En la actualidad, el enfoque más prometedor parece ser la creación de aplicaciones basadas en HTML5 con JavaScript.

**Figura 2**




**Evolución de las ventas de los sistemas operativos para dispositivos móviles. Fuente: http://www.businessinsider.com/mobile-platform-market-share-2012-8.**

## ENFOQUES PARA EL DESARROLLO DE APLICACIONES HTML5 INDEPENDIENTES DE LA PLATAFORMA

Podemos decir que existen principalmente dos enfoques diferentes para desplegar (deploy) y correr aplicaciones basadas en la Web con HTML5 dependiendo de las condiciones del ambiente. La primera, la más obvia, es usar un navegador Web que entienda HTML5 y simplemente ingresar la URL del sitio (página) que contenga el código y JavaScript, el cual será interpretado. Este enfoque bien general implica que la aplicación no tendrá acceso a los accesorios específicos del dispositivo móvil, pues el código JavaScript no tiene cómo acceder a ellos, ya que son específicos de cada máquina (excepción a esto es el acceso a los datos de geolocalización, como se vio más arriba). Esto es así dado el carácter general y portable que debe tener el código escrito en JavaScript.

Un segundo enfoque para el despliegue de aplicaciones basadas en HTML5, es usar una aplicación nativa especialmente desarrollada para la plataforma que se desea usar, que contenga un motor HTML5, es decir, una funcionalidad capaz de bajar una página de un servidor e interpretarla, y que tenga acceso a los accesorios del dispositivo. Obviamente, esta aplicación no será totalmente portable a otra plataforma. Un enfoque que se perfila como el más conveniente cuando se necesita que una aplicación converse con los accesorios del dispositivo, es usar el segundo enfoque pero programar lo más que se pueda en HTML, de modo que al momento de portar la aplicación se tenga que reimplementar lo menos posible.

## BIBLIOTECAS ABIERTAS EN HTML5

Durante los últimos meses han aparecido algunas bibliotecas de JavaScript que facilitan el desarrollo de aplicaciones más complejas en HTML, algunas de ellas especialmente orientadas a trabajar con los elementos de HTML5. A continuación describimos algunas de ellas:

**JQuery:** es quizá la herramienta más popular para trabajar con JavaScript. Si bien no es exclusiva de HTML5, es ampliamente usada para trabajar en la Web. El núcleo de JQuery consiste en la redefinición del símbolo $ del lenguaje para que éste contenga una serie de funciones que se enfocan en hacer fácil lo que con JavaScript era complejo de hacer. Por ejemplo, la selección de elementos de HTML (DOM) se realiza de una manera práctica e intuitiva, incorporando una estructura de consultas sobre los elementos que permite retornar una o más coincidencias. En particular, podemos seleccionar todos los links de un documento y manipularlos, o sólo los links que se encuentren dentro de una lista. JQuery además provee de utilitarios que simplifican las operaciones que son más repetitivas, como llamadas AJAX, conversión del formato de retorno (texto, XML, JSON) a objetos manipulables y otras. Gracias a JQuery actualmente vemos avanzadas interfaces gráficas en la Web, que nos permiten incluso editar imágenes en tiempo real sin tener que subir datos al servidor.

> Podríamos decir que HTML5 se diferencia de su predecesor HTML4 en cuanto a que este último incorpora una serie de elementos nuevos que pueden ser instanciados y manipulados desde pedazos de código JavaScript insertos en la página web.

**Modernizr:** en la actualidad tenemos un sinfín de navegadores con formas diferentes de interpretar HTML/HTML5. Muchos de estos soportan operaciones de manera parcial lo que hace complejo diseñar una aplicación sin saber dónde se ejecutará. Una alternativa es generar múltiples versiones de nuestra aplicación y entregar la que corresponde al navegador. La otra es usar Modernizr, una librería que se encarga de detectar la disponibilidad de HTML5 en el explorador y según esto se encarga de incorporar código en demanda de manera dinámica y a la medida. Esto permite que un mismo documento se pueda comportar de manera diferente según las capacidades del navegador.

**Kendo / Ext js:** al diseñar un sistema, comúnmente se definen qué componentes se utilizarán, cómo se manejará si existen múltiples idiomas, se definen las interfaces gráficas y, en el caso de utilizar un modelo vista controlador, corresponde definir cada uno también. Kendo y Ext js son recopilaciones de todo lo anterior para JavaScript. Es decir, de manera fácil se puede especificar el soporte para múltiples idiomas, componentes gráficos, el "tema" de la "interface" los modelos, vistas y

controladores de la aplicación. La ventaja es que, dado que se ejecuta en el navegador, no existe sobrecarga de los servidores para la gestión de la aplicación, lo cual permite tener plataformas escalables con el mismo o menor esfuerzo que antes.

**PhoneGap:** debido a que el manejo de los accesorios de multimedia y posicionamiento de los dispositivos móviles es muy dependiente de su tipo, se han desarrollado bibliotecas que tienden a uniformizar este manejo para hacer así más portable el código. La API más utilizada para este propósito externo es probablemente PhoneGap, compatible con al menos cuatro de los cinco principales sistemas operativos para dispositivos móviles. El único que falta aquí es Windows Mobile, que los desarrolladores de PhoneGap se comprometieron a integrar en el futuro también. Por desgracia, la API PhoneGap no permite el acceso a todos y cada dispositivo de hardware depende del sistema operativo móvil (la Tabla 1 muestra qué tipo de hardware está soportado bajo qué sistema operativo móvil). Sin embargo, PhoneGap proporciona suficiente apoyo de manejo de accesorios específicos a una plataforma para los sistemas operativos móviles habituales, de manera de apoyar el desarrollo de aplicaciones independientes de la plataforma para variados escenarios de la computación móvil.

## PERSPECTIVAS

Al revisar con detención las características de HTML5 y las posibilidades que ofrece para el desarrollo de aplicaciones altamente interactivas y complejas corriendo dentro de un navegador Web, el desarrollador que ha visto la evolución de las aplicaciones basadas en la Web se preguntará qué hay de diametralmente nuevo en este enfoque que no se haya visto aún. En efecto, si el lector ya desarrollaba aplicaciones basadas en la Web hacia finales de los noventa recordará que HTML5 tiene muchos elementos que son característicos de los Applets de Java. Sin embargo, hoy vemos que son pocas las páginas que integran esta tecnología a pesar de los grandes augurios con la que fue recibida en su momento.

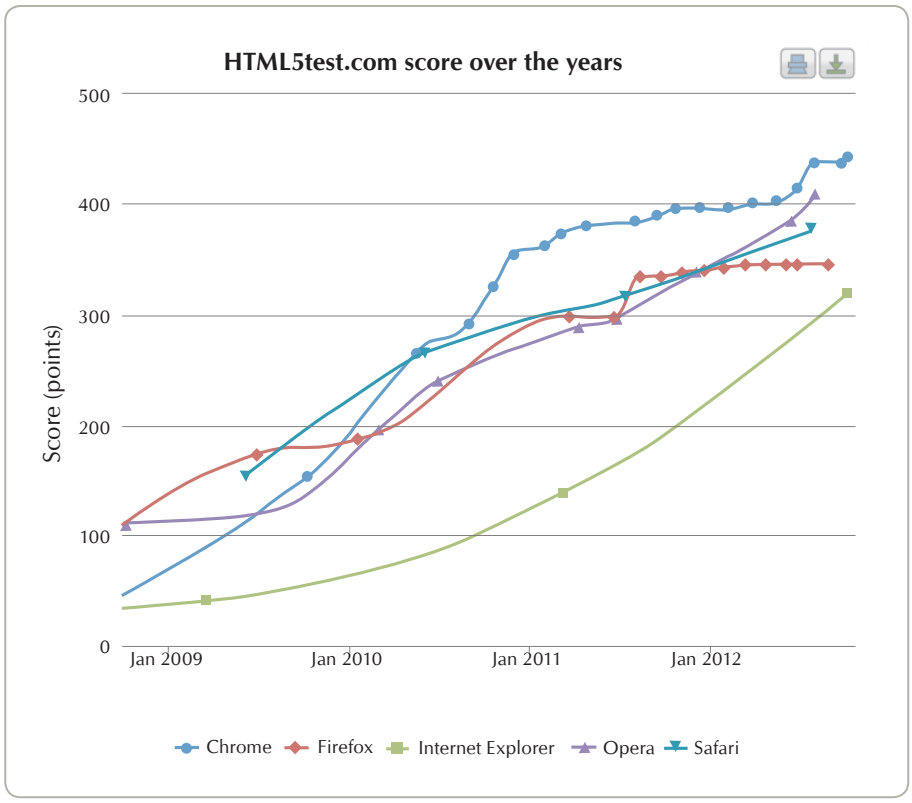
Quizás para preguntarse qué perspectivas tiene HTML5 sea necesario también analizar cuáles fueron las razones por las cuales los Applets de Java no llegaron a ser el estándar que se pensaba llegarían a ser. Ciertamente en todo establecimiento de un estándar los intereses económicos de las compañías juegan un papel importante. Java fue desarrollado y mantenido por una compañía específica (Sun Microsystems) y si bien fue liberado sin costo, podríamos pensar que la intención detrás era que los productos de otras compañías no se impusieran (por lo tanto, habían intereses económicos de por medio entre proveedores del mercado). En este sentido HTML5 comienza con mejor perspectiva ya que no fue desarrollado ni "apadrinado" por ninguna compañía: las versiones de HTML son sancionadas y liberadas por la organización W3C, que se encarga de establecer los estándares para la Web, por lo que podríamos pensar que su establecimiento como estándar estará relativamente libre de intrigas. Sin embargo, también hubo factores técnicos que quizás pesaron más aún que los económicos que jugaron en contra de los Applets y que hasta cierto punto se da también en la adopción de HTML5 como estándar. Esto tiene que ver con la velocidad con que los navegadores incorporan las capacidades para interpretar completamente el código. Hacia mediados de los noventa se libera la primera versión de Java y muchos de los navegadores de la época incorporan la máquina virtual de Java (JVM) capaz de interpretar los Applets, que son desarrollados con la primera versión de Java. Sin embargo, Java se desarrolla muy rápido y los Applets desarrollados con las versiones posteriores no pueden ser interpretados por la máquina virtual de versiones anteriores. En esos tiempos la autoactualización del software, tal como ocurre hoy, no estaba implementada en los navegadores. Esto causó entonces

**Tabla 1**

| | Android | iOS | Symbian | Blackberry | | Android | iOS | Symbian | Blackberry |
|---|---|---|---|---|---|---|---|---|---|
| Acelerómetro | x | x | x | x | Notificación | x | x | x | x |
| Cámara | x | x | x | x | Geolocalización | x | x | x | x |
| Brújula | x | x | - | - | Multimedia | x | x | - | - |
| Lista de contactos | x | x | x | x | Red | x | x | - | x |
| Información del dispositivo | x | x | - | x | Sistema de archivo | x | x | - | x |
| Eventos | x | x | - | x | Almacenamiento | x | x | - | x |

**Soporte para el manejo de accesorios específicos del hardware por sistema operativo que brinda PhoneGap.**

**Figura 3**




**Curva de implementación de HTML5 para los principales navegadores. La medición (score) supone un máximo de 500 puntos para un 100% de implementación de las funcionalidades definidas por el estándar.**

que muchos sitios que integraban Applets desarrollados con las últimas versiones no pudiesen verse en un porcentaje importante de los navegadores.

Si bien el problema de los navegadores con los Applets fue el no contar con la versión correcta de la máquina virtual de Java, hoy en día vemos que la implementación de todas las funcionalidades que define el estándar por parte de los navegadores ha sido más bien lenta. En el sitio http://html5test.com, es posible medir en qué medida el browser que contacta el servidor del sitio tiene implementadas las funcionalidades de HTML5. En esta misma página se puede ver también un gráfico que muestra cómo ha sido la evolución en la adopción de las características de HTML5 por parte de los navegadores más comunes de hoy, esto es, en qué medida los navegadores son capaces de entender el protocolo HTML5 e implementar las funcionalidades que se requieren para interpretarlo (ver Figura 3). Si bien la tendencia ha sido positiva, se han requerido tres años para llegar a una situación donde la mayoría de los navegadores ha implementado más del 60% de las funciones del estándar. Sólo dos superan el 80% y ninguno el 90%.

Otro problema que también atenta fuertemente contra el establecimiento de HTML5 como un estándar, es el hecho de que los resultados que se obtienen en cuanto a la interpretación del código varían de navegador a navegador, e incluso entre distintas versiones de un mismo navegador, especialmente entre las versiones para dispositivos móviles y para desktop. Por ejemplo, Chrome utiliza un webkit que provee algunas funcionalidades, tales como ver imágenes captadas por la cámara, pero no permite grabar de la misma. Lo mismo ocurre con el sonido. Las llamadas para controlar estos accesorios de multimedia son diferentes en Android, Mozilla y Opera. Por ahora, Opera es el único que respeta el estándar, tanto la versión para dispositivos móviles como en la de dispositivos de escritorio (desktop). Los navegadores Firefox e Internet Explorer implementan sólo las funcionalidades más populares: WebDatabase, LocalStorage, FileSystem y Canvas. Websockets funcionan en la versión desktop de Chrome pero no en móvil. Lo mismo sucede con Firefox. Chrome incorporó el sonido hace poco en la versión 2.2, en las anteriores no estaba disponible. Por esto es que existen librerías que autodetectan todas las características implementadas y autorrenombran las llamadas para que se ajusten al estándar.

Del análisis anterior podemos concluir que HTML5 aún no está en condiciones de ser nombrado "el" estándar para estos tiempos. En el caso de los Applets, cuando se quiso presentarlos como una solución estándar para varios problemas que había en ese entonces, las variadas dificultades e inconsistencias que mostraban los navegadores al momento de interpretarlos terminaron por hacer que los desarrolladores de páginas web evitaran incorporarlos en sus páginas. Lo curioso es que hoy en día es probable que los Applets puedan funcionar sin problemas y sin diferencias en casi cualquier navegador. Esto se debe a que Java está mucho más maduro y estable, y por lo tanto ya no evoluciona de manera tan dinámica, y además es normal que los navegadores se autoactualicen bajando nuevas versiones y parches casi sin que los usuarios se den cuenta. Es probable que de aquí a un tiempo no muy lejano la interpretación que los navegadores hagan de HTML5 también sea más completa y estandarizada incluso para las versiones móviles y desktop. Cabe entonces hacerse la pregunta de si para ese entonces los desarrolladores de sitios web habrán decidido ya evitar esta tecnología y pase algo parecido a lo sucedido con los Applets o si para entonces ya habrá disponible otra tecnología que prometa (¿de nuevo?) ser el estándar del futuro de la Web.BITS