DESARROLLO WEB ENTORNO CLIENTE

Desarrollo de Aplicaciones Web

Tema 1

Arquitecturas y lenguajes de programación en clientes web

ÍNDICE

Arquitecturas y lenguajes de programación en clientes web.

1. Desarrollo web.

La web fue inicialmente concebida y creada por **Tim Berners-Lee**, un especialista del laboratorio europeo de partículas (CERN) en 1989. En sus mismas palabras, había una "necesidad de una herramienta colaborativa que soportara el conocimiento científico" en un contexto internacional. Él y su compañero Robert Cailliau crearon un prototipo web para el CERN y lo mostraron a la comunidad para sus pruebas y comentarios.

Dicho prototipo estaba basado en el concepto de hipertexto (*Texto que cuando pulsamos en él nos conduce a otrotexto, objeto, sonido, video, sección o documento relacionado*). Como resultado se crearon unos protocolos (*cuando pulsamos en él nos conduce a otro texto, objeto, sonido, video, sección o documento relacionado*) y especificaciones que han sido adoptados universalmente e incorporados a Internet, gracias a aportaciones posteriores como el desarrollo por la NCSA de la popular interfaz MOSAIC.

Todos los prototipos y desarrollos posteriores crecieron bajo la guía del **consorcio W3C**, que es una organización con base en el MIT de Massachusetts y que se responsabiliza de desarrollar y mantener los estándares web.

Por **Web se pueden entender tres cosas distintas:** el proyecto inicial del CERN, el conjunto de protocolos desarrollados en dicho proyecto o bien el espacio de información formado por todos los servidores interconectados. Cuando se habla de la Web generalmente se hace referencia a esto último.

Muchas de las discusiones sobre Diseño Web o Desarrollo Web son confusas, ya que la expresión varía considerablemente. Mientras que la mayoría de la gente tiene algún tipo de noción sobre **lo que significa Diseño Web**, solamente unos pocos son capaces de definirlo con exactitud, y tú vas a estar dentro de ese grupo.

Algunos componentes como diseño gráfico o programación, forman parte de esa discusión, pero su importancia en la construcción de webs varía de persona a persona y de web a web. Algunos consideran la creación y organización de contenido - o más formalmente, la arquitectura de la información - como el aspecto más importante del Diseño Web. Otros factores como - la facilidad de uso, el valor y funcionalidad del sitio web en la organización, su funcionalidad, accesibilidad, publicidad, etc. También forman una parte muy activa hoy en día sobre lo que se considera Diseño Web.

El Desarrollo Web ha sido y sigue estando muy influenciado por múltiples campos como el de las nuevas tecnologías, los avances científicos, el diseño gráfico, la programación, las redes, el diseño de interfaces (Medio o forma a través de la cuál un usuario se comunica con el ordenador) de usuario, la usabilidad y una variedad de múltiples recursos. Por lo tanto el Desarrollo Web es realmente un campo multidisciplinar.

1.1. Áreas

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.

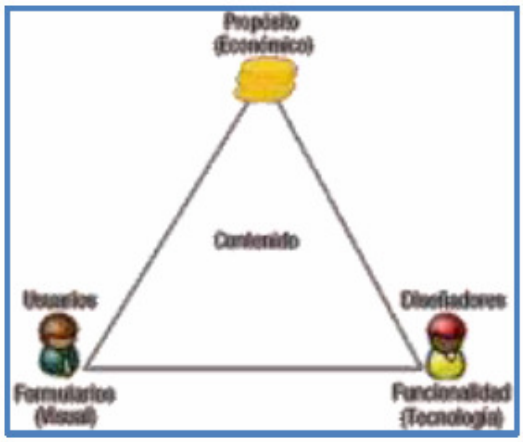
- ✓ **Tecnología**: aunque muchas de las tecnologías web como HTML o CSS entran dentro de esta categoría, la tecnología en este contexto generalmente hace referencia a los diferentes tipos de elementos interactivos de un sitio web, generalmente aquellos construidos empleando técnicas de programación.

- ✓ **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(server-side)**: incluye el hardware y software del servidor Web así como diferenteselementos de programación y tecnologías incrustadas. Las tecnologías pueden abarcar un rango amplio desde programasCGI escritos en PERL hasta aplicaciones multihilo (*También denominado multiproceso hace referencia a la posibilidad de ejecutar diferentes trozos de código de una misma aplicación de forma simultánea*) basadas en Java, incluyendo tecnologías de servidor de bases de datos que soporten múltiples sitios web.

- ✓ El lado del **cliente(client-side)**: este elemento 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. Es justamente aquí dónde nos vamos a centrar a lo largo de todo el módulo.

- ✓ La **red**: describe los diferentes elementos de conectividad (*Capacidad que tiene un dispositivo para poder* conectarse a otros. Aquí se detallan los diferentes protocolos y material utilizado para poder realizar dicha conexión) utilizados para mostrar el sitio web al usuario.

El entendimiento completo de todos los aspectos técnicos del medio Web, incluyendo la componente de red, es de vital importancia para llegar a ser un buen Diseñador Web.

2. Lenguajes de programación en clientes web.

Cuando hablamos de tecnologías empleadas en lenguajes de programación web podemos citar dos grupos básicos: **client-side** y **server-side**. Las tecnologías client-side 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 server-side.

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 client-side y server-side 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 (server-side), 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).

En este módulo nos vamos a centrar principalmente en estos últimos, los lenguajes de scripting, y en particular en el lenguaje **JavaScript** que será el lenguaje que utilizaremos a lo largo de todo este módulo formativo.

Tabla comparativa de lenguajes de programación web cliente – servidor.

Lado del Cliente (client-side)	Lado del servidor (server-side)
✓ 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 servidor. ➔ Módulos de Apache. ➔ Extensiones ISAPI y filtros. ➔ Servlets de Java. ✓ Lenguajes de scripting. ➔ PHP. ➔ Active Server Pages (ASP/ASP.NET). ➔ ColdFusion.
...	

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 mayoritariamente por todas las plataformas (*Sistema operativo utilizado por un determinado dispositivo*). Por lo tanto a partir de ahora todas las referencias que hagamos estarán enfocadas hacia JavaScript.

A continuación te mostramos un esquema de las **4 capas del desarrollo web** en el lado del cliente, en la que se puede ver que JavaScript se sitúa en la capa superior gestionando el comportamiento de la página web.

Tabla de las 4 capas del desarrollo web en el lado del cliente.	
Comportamiento (JavaScript)	
Presentación (CSS)	
Estructura (DOM / estructura HTML)	Contenido Estructurado (documento HTML)
Contenido (texto, imágenes, vídeos, etc.)	

2.1. Características.

Como vimos anteriormente, 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:

- ✓ **Primero**: 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.
- ✓ **Segundo**: 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 ésto 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.
- ✓ **Tercero**: 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 (http://maps.gogle.com).

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 (*Programa informático capaz de analizar y ejecutar código escrito en un lenguaje de programación de alto nivel (aquel que se acerca más a la capacidad cognitiva de las personas en lugar de a la capacidad ejecutora de la máquina*) 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.

2.2. Compatibilidades.

A diferencia de otros tipos de scripts como los CGI, 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.

Navegadores actuales y su soporte para distintas tecnologías

Browser	JavaScript	ECMAScript 3	DOM 1	DOM 2	DOM 3	XPath	DHTML	XMLHttpRequest	Rich editing
Amaya	No [note 1]	No [note 1]	No [note 1]	No	No	No	No	No	No
AOL Explorer	Yes	Yes	Partial	Yes	No	No	Yes	Yes	Yes
Avant	Yes	Yes	Partial	No	No	No	Yes	Yes	Yes
Camino	Yes	Yes	Yes	Yes	No [note 2]	Yes	Yes	Yes	Yes
Dillo	No	No	No	No	No	No	No	No	No
DocZilla	Yes	No	Yes	Yes	Partial	Yes	Yes	Yes	No
ELinks	Partial	Partial	No	No	No	No	No	No	No
Web	Yes	Yes	Yes	Yes	No [note 2]	Yes	Yes	Yes	Yes
Flock	Yes	Yes	Yes	Yes	No [note 2]	Yes	Yes	Yes	Yes
Galeon	Yes	Yes	Yes	Yes	No [note 2]	Yes	Yes	Yes	Yes
Google Chrome	Yes	Yes	Yes	Yes	Partial	Yes	Yes	Yes	Yes
iCab	Yes	Yes	Partial	Partial	No	No	Yes	Yes	No
Internet Explorer	Yes	Yes	Yes	Yes [note 3]	Partial	Yes	Yes	Yes	Yes
Internet Explorer for Mac	Yes	Yes	Partial	No	No	No	Yes	No	No
K-Meleon	Yes	Yes	Yes	Yes	No [note 2]	Yes	Yes	Yes	Yes
Konqueror	Yes	Yes	Yes	Yes	Partial	No	Yes	Yes	No
Links	No [note 4]	No	No	No	No	No	No	No	No
Lynx	No	No	No	No	No	No	No	No	No
Maxthon	Yes	Yes	Partial	No	No	Yes	Yes	Yes	Yes
Midori	Yes	Yes	Yes	Yes	Partial [note 5]	Yes [note 5]	Yes	Yes	Yes
Mosaic	No	No	No	No	No	No	No	No	No
Mozilla	Yes	Yes	Yes	Yes	No [note 2]	Yes	Yes	Yes	Yes
Mozilla Firefox	Yes	Yes	Yes	Yes	Partial	Yes	Yes	Yes	Yes
Netscape	Yes	Yes	Yes	Yes	No [note 2]	Yes	Yes	Yes	Yes
Netscape Browser	Yes	Yes	Depends [18]	Depends [18]	No [note 2]	Depends [18]	Yes	Yes	Yes
Netscape Navigator	Yes	Partial	No	No	No	No	Yes	No	No
Netscape Navigator 9	Yes	Yes	Yes	Yes	No [note 2]	Yes	Yes	Yes	Yes
NetSurf	No	No	No	No	No	No	No	No	No
OmniWeb	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No
Opera	Yes	Yes	Yes	Yes	Partial	Yes	Yes	Yes	Yes
Safari	Yes	Yes	Yes	Yes	Partial [note 5]	Yes	Yes	Yes	Yes
SeaMonkey	Yes	Yes	Yes	Yes	No [note 2]	Yes	Yes	Yes	Yes
Shiira	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes
Sleipnir	Yes	Yes	Partial	No [note 3]	No	Yes	Yes	Yes	Yes
WorldWideWeb	No	No	No	No	No	No	No	No	No
w3m	No	No	No	No	No	No	No	No	No

A veces las incompatibilidades entre navegadores al interpretar el código de JavaScript no vienen dadas por el propio código en si, 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: http://validator.w3.org/

También tienes 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.

2.3. 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.

3. Herramientas y utilidades de programación (I).

La mejor forma de aprender JavaScript es tecleando el código HTML y JavaScript en un simple documento de texto. La elección del editor depende de ti, pero aquí te voy a dar algunas pistas para realizar una buena elección.

Para aprender JavaScript no se recomiendan editores del estilo **WYSIWYG** (What You See is What You Get) como Dreamweaver o FrontPage, ya que estas herramientas están más orientadas a la modificación de contenido y presentación, y nosotros nos vamos a centrar más en el código fuente de la página.

Uno de los **factores** importantes que tienes que tener en cuenta a la hora de elegir un editor, es ver la facilidad con la que se pueden grabar los ficheros con extensión .html. Independientemente del sistema operativo que estés utilizando cualquier programa que te permita grabar ficheros directamente con la extensión .htm o .html te evitaría un gran número de problemas. También hay que tener en cuenta la codificación que emplea ese programa para grabar los ficheros. Por ejemplo en Microsoft Word cuando intentamos guardar archivos éste intenta almacenarlos en un formato binario de Word - algo que los navegadores web no pueden cargar. Para grabar ese archivo con extensión .txt o .html requiere moverse un poco más por el menú de diálogo "Guardar como", lo cuál es realmente una clara desventaja.

La idea es decantarse por editores que posean **características** que te faciliten la programación web, como por ejemplo:

- ✓ Sintaxis con codificación de colores. Que resalte automáticamente en diferente color o tipos de letra los elementos del lenguaje tales como objetos, comentarios, funciones, variables, etc.
- ✓ Verificación de sintaxis. Que te marque los errores en la sintaxis del código que estás escribiendo.
- ✓ Que te permita diferenciar los comentarios del resto del código.
- ✓ Que genere automáticamente partes del código tales como bloques, estructuras, etc.

- ✓ Que disponga de utilidades adicionales, tales como cliente FTP para enviar tus ficheros automáticamente al servidor, etc.

Dependiendo del sistema operativo que utilices en tu ordenador dispones de múltiples **opciones de editores**. Cada una es perfectamente válida para poder programar en JavaScript. Algunos ejemplos de editores web gratuitos son:

- ✓ Para Windows tienes: Notepad++, Aptana Studio, Bluefish, Eclipse, NetBeans, etc.
- ✓ Para Macintosh tienes: Aptana Studio, Bluefish, Eclipse, KompoZer, Nvu, etc.
- ✓ Para Linux tienes: KompoZer, Amaya, Quanta Plus, Bluefish, codetch, etc.

Otro de los componentes obligatorio para aprender JavaScript es el navegador web. No es necesario que te conectes a Internet para comprobar tus scripts realizados con JavaScript. Puedes realizar dicha tarea sin conexión. Ésto quiere decir que puedes aprender JavaScript y crear aplicaciones con un ordenador portátil y desde cualquier lugar sin necesitar Internet para ello.

3.1. Herramientas y utilidades de programación (II).

El tipo de **navegador web** que utilices es elección tuya. Eso sí, te recomiendo que uses las últimas versiones disponibles para evitar problemas de seguridad e incompatibilidades. Algunos ejemplos de navegadores gratuitos son:

- ✓ Para Windows tienes: Mozilla Firefox, Google Chrome, Safari, Opera, Internet Explorer, etc.
- ✓ Para Macintosh tienes: Mozilla Firefox, Safari, Google Chrome, Internet Explorer, etc.
- ✓ Para Linux tienes: Mozilla Firefox, Konqueror, Opera, etc.

Una recomendación muy interesante es el disponer de 2 o 3 tipos de navegadores diferentes, ya que así podrás comprobar la compatibilidad de tu página web y ver si tu código fuente de JavaScript se ejecuta correctamente en todos ellos.

Para ajustar un poco más tu entorno de trabajo, lo último que necesitas es el poder ejecutar tu editor web y tu navegador de forma simultánea, ya que el **flujo típico de trabajo en JavaScript** va a ser:

1. Introducir HTML, JavaScript y CSS en el documento original en el editor web.
2. Guardarlo en disco.
3. Cambiarte al navegador web.
4. Realizar una de las siguiente tareas:

- ✓ Si es un nuevo documento, abrirlo a través de la opción Abrir del menú Archivo > Abrir Archivo.
- ✓ Si el documento ya está cargado en el navegador pues simplemente recargar la página.

Los pasos 2 al 4 son acciones que se van a ejecutar muy frecuentemente. Esa secuencia grabar-cambiar-recargar la realizarás tantas veces cuando estés escribiendo y depurando tu script, que llegará a ser prácticamente un acto reflejo. Algunos editores ya disponen de teclas rápidas para realizar esta tarea. Todo ello dependerá del tipo de editor, navegador y sistema operativo que utilices.

Otro aspecto muy importante es la **validación**. Puedes ahorrarte muchas horas de comprobaciones simplemente asegurándote de que tu código HTML es válido. Si tu código HTML contiene imperfecciones, tienes muchas posibilidades de que tu JavaScript o CSS no funcionen de la manera esperada, ya que ambos dependen de los elementos HTML y sus atributos. Cuanto más te ajustes a las especificaciones del estándar, mejor resultado obtendrás entre los diferentes tipos de navegadores.

El consorcio W3C, el cuál diseño el lenguaje HTML, desarrolló un validador que te permitirá chequear si tu página web cumple las especificaciones indicadas por el elemento DOCTYPE que se incluye al principio de cada páginaweb que realices. El Validador será tu amigo y te permitirá realizar unas páginas más consistentes.

La dirección del **Validador W3C** es: http://validator.w3.org/

Ofrece tres formas de introducción de tu código para la validación - copiando y pegando tu código enun formulario, enviando el fichero .html o bien indicando la dirección URL dónde se encuentra nuestra página web. A continuación se pulsa el botón de chequear y el validador nos devolverá los errores encontrados. Realizaremos los cambios necesarios y procederemos a validar de nuevo hasta que no tengamos más errores.

Para más información sobre las diferentes especificaciones HTML y CSS.
http://www.w3.org/TR/html4/
http://www.w3.org/TR/html5/
http://www.w3.org/TR/xhtml1/
Especificación sobre:
http://www.w3.org/TR/CSS21/
http://jigsaw.w3.org/css-validator/

4. Integración de código Javascript con HTML (I).

Ahora que ya conoces las herramientas que puedes utilizar para comenzar a programar en JavaScript, vamos a ver la forma de **integrar el código de JavaScript** en tu código 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 deJavaScript 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 `js/`)

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.

Seguramente estarás pensando en cómo puedes **proteger el código de JavaScript** que vas a programar, del uso fraudulento por otros programadores o visitantes a tu página: la respuesta rápida a esa pregunta es que es imposible hacerlo.

Para que el código de JavaScript pueda ejecutarse correctamente deberá ser cargado por el navegador web y por lo tanto su código fuente deberá estar visible al navegador. Si realmente te preocupa que otras personas usen o roben tus scripts, deberías incluir un mensaje de copyright en tu código fuente. Piensa que no solamente tus scripts son visibles al mundo, sino que los scripts del resto de programadores también lo son. De esta forma puedes ver fácilmente cuando alguien está utilizando al pie de la letra tus scripts, aunque esto no evita que alguien copie tu código y elimine tu mensaje de copyright.

Lo más que se puede hacer es ofuscar el código, o hacerlo más difícil de entender. Las técnicas de ofuscación incluyen la eliminación de saltos de línea, espacios en blanco innecesarios, tabuladores, utilización de nombres ininteligibles en las funciones y variables, utilización de variables para almacenar trozos de código, uso de recursividad, etc. La forma más rápida de hacer todas esas tareas de ofuscación es utilizar un software que producirá una copia "comprimida" (*Reducir el espacio físico que ocupa el código fuente de nuestra aplicación, sin que se produzca pérdida de información*) del script que has programado para facilitar su carga rápida.

Para que veas un ejemplo de ofuscador de JavaScript visita: http://www.javascriptobfuscator.com/

Lo mejor es que cambies ese paradigma y pienses de una manera diferente. En lugar de proteger tu código, lo mejor es promocionarlo y hacer ostentación de él, añadiendo comentarios útiles en el código fuente, publicándolo en tu blog o en webs de programación, etc. Incluso podrás añadir una licencia **Creative Commons** para animar a la gente a que lo utilice, lo copie y lo mantenga público al resto del mundo. Así conseguirás mayor reputación como buen programador y la gente contactará contigo para más información, posibles trabajos, etc.
Para más información: http://es.creativecommons.org/licencia/