Tema 1 Arquitecturas y lenguajes de programación en clientes web

# 1.- Desarrollo web.

Tim Berners-Lee 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. Como resultado, se crearon unos protocolos 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 consorcioW3C (organización con base en el MIT de Massachusetts**).

**Por web se pueden entender 3 cosas distintas**, generalmente se entiende por la última:

* El proyecto inicial del CERN.
* El conjunto de protocolos desarrollados en dicho proyecto.
* **El espacio de información formado por todos los servidores interconectados**.

Muchas de las discusiones sobre Diseño Web o Desarrollo Web son confusas, ya que la expresión varía considerablemente.

Algunos componentes como diseño gráfico o programación, forman parte de esta discusión. **Otros, consideran la creación y organización de contenido (la arquitectura de la información) como el aspecto más importante del Diseño web**. Otros factores son la facilidad de uso, el valor y funcionalidad del sitio web en la organización, su accesibilidad, la publicidad, etc.

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](https://ead.murciaeduca.es/pluginfile.php/1257131/mod_resource/mod/glossary/showentry.php?displayformat=dictionary&concept=Interfaz%20%28DAW_DWEC01%29) de usuario, la usabilidad y una variedad de múltiples recursos.

## 1.1.- Áreas.

* **Contenido**: es la **forma y organización del contenido del sitio**. Se presenta organizado y estructurado utilizando tecnologías de marcas.
* **Visual**: plantilla empleada en un sitio web. Esta plantilla generalmente se genera usando HTML y CSS y puede incluir elementos gráficos para decoración o para navegación.
* **Tecnología**: HTML, CSS, distintos tipos de elementos interactivos (Javascript)…
* **Distribución**: velocidad y fiabilidad con la que un sitio web se distribuye., en función del 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 para interés económico.

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.

Una forma de pensar en los componentes del Diseño web es mediante la **estructura de pirámide**: 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 en la mayoría de casos.

Hoy en día, los sitios web **siguen un modelo basado en la programación cliente-servidor** basado en 3 elementos comunes:

* **Lado del servidor (server-side)**: incluye el hardware y el software del servidor Web.
  + **Tecnologías**: desde programas CGI escritos en PERL hasta aplicaciones multihilo basadas en Java, incluyendo tecnologías de servidor de bases de datos.
* **Lado del cliente (client-side)**: hace referencia a los navegadores web.
  + **Tecnologías**: HTML, CSS, JavaScript, controles ActiveX. Se utilizan para crear la presentación de la página o proporcionar características interactivas.
  + **La red**: describe los diferentes elementos de conectividad utilizados para mostrar el sitio web al usuario.

# 2.- Lenguajes de programación en clientes web.

Uno de los objetivos en la programación web es saber **escoger la tecnología correcta** para tu trabajo.

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. Por otro lado el almacenar los datos en el servidor estaría mucho mejor gestionado por una tecnología del lado del servidor (server-side).

La mezcla es generalmente la mejor solución.

**Cuando hablamos de lenguajes de programación en clientes web, podemos distinguir dos variantes**:

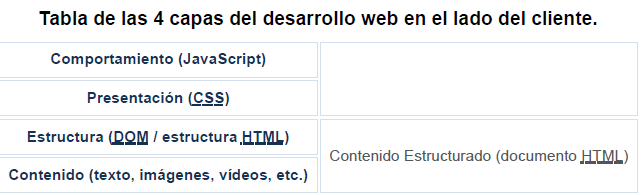
* Lenguajes que nos permiten dar formato y estilo a una página web (HTML, CSS, etc.)
* Lenguajes que nos permiten aportar dinamismo a páginas web (lenguajes de scripting).
  + En este módulo nos centraremos en estos.

**Tabla comparativa de los 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. |

Elegimos Javascript en este módulo porque es el lenguaje de script más utilizado en la programación en el lado del cliente, y está soportado mayoritariamente por todas las plataformas.

Esquema de las 4 capas del desarrollo web en el lado del cliente:



## 2.1.- Características.

**Los lenguajes de programación para clientes web no son un reemplazo de la programación en el lado del servidor**. Las razones son simples:

1. **JavaScript** por sí mismo **no puede escribir ficheros en el servidor**.
2. **No todos los clientes web ejecutan JavaScript**.
3. Uno de los caminos que más ha **integrado la programación cliente con la programación servidor** ha surgido gracias a **AJAX**.
   1. **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.
   2. Ejemplo de ello: Google Maps funciona así.

**JavaScript está orientado a dar soluciones a**:

* Conseguir que nuestra página web responda 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.
* **Preprocesar datos** en el cliente antes de enviarlo 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 programación.

**Los lenguajes de script no se usan solamente en las páginas web**. Los intérpretes de JavaScript están integrados en múltiples aplicaciones de uso cotidiano. **Por ejemplo**: Google Desktop Gadgets, Adobe Acrobat, Dreamweaver, OpenOffice.org, Google Docs, etc.

## 2.2.- Compatibilidades.

**JavaScript es interpretado por el cliente** (navegador). Por ejemplo: Firefox, Chrome, Safari, Opera, Internet Explorer, etc.

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.

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

**A veces las incompatibilidades entre navegadores** al interpretar el código de JavaScript no **vienen dadas por** el propio código en sí, sino que su origen proviene **del código fuente HTML**.

[Enlace al validador W3C HTML](https://validator.w3.org/)

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

* **No todos los navegadores soportan lenguajes de script** 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**: existen 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. Para evitar esto, los navegadores web en el cliente aplican dos tipos de restricciones:

* **Ejecución del código en un “espacio seguro de ejecución”** en el cual solamente podrá realizar tareas relacionadas con la web, nada de tareas genéricas de programación.
* **Los script están restringidos por la política de “mismo origen”**: los scripts de una web no tendrán acceso a información tal como usuarios, contraseñas, o cookies enviadas desde otra web.

Además, existen **limitaciones** de **JavaScript** **que** en parte **refuerzan sus capacidades de seguridad**. **No puede**:

* **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 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 web.
* **Interactuar directamente con los lenguajes de servidor**.
* **No puede acceder a páginas web almacenadas en diferentes dominios**.
* **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”**. Ocurre cuando un atacante es capaz de **inyectar código malicioso en la página web presentada a su víctima**.

**Otro aspecto de 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 S.O del usuario.

El **motor de ejecución de JavaScript** es el encargado de ejecutar el código de JavaScript en el navegador y en el que recae el peso fuerte de la implementación de la seguridad.

* **Active Script de Microsoft**: tecnología JScript como lenguaje de scripting. A veces se considera compatible con JavaScript, pero Microsoft emplea múltiples características que no siguen los estándares ECMA.
* **El kit de herramientas Qt C++** .
* **El lenguaje de programación Java**, que en su versión JDK 1.6 introduce un paquete denominada javax.script
* Y por supuesto todos los motores implementados por los navegadores web, cada uno de ellos da soporte a alguna de las diferentes versiones de JavaScript.

Hoy en día una de las **características que diferencia 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 cada una de las pestañas de navegación.

# 3.- Herramientas y utilidades de programación (I).

**La elección del editor depende de ti**. 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.

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**

**Cualquier programa que te permita grabar ficheros directamente con la extensión .htm o .html serviría**, aunque programas como Word no son la mejor opción para esto, ya que hay que moverse por el diálogo de Guardar como, lo cual es una clara desventaja.

También hay **que tener en cuenta la codificación** que empleamos para grabar los ficheros.

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

* **Sintaxis con codificación de colores**. Resaltado en diferente color o tipo de letra los elementos del lenguaje (objetos, comentarios, funciones, variables, etc.).
* **Verificación de sintaxis**: que te marque los errores de sintaxis en el código.
* **Diferenciación los comentarios del resto del código**.
* Generación automática de partes del código como bloques, estructuras, etc.
* **Que disponga de utilidades adicionales**: cliente FTP para enviar tus ficheros automáticamente al servidor, etc.

**Ejemplos de editores web gratuitos**:

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

**Otro de los componentes obligatorio** para aprender JavaScript **es el navegador web**, ya que no es necesario conexión a Internet para comprobar tus scripts realizados con JavaScript.

## 3.1.- Herramientas y utilidades de programación (II).

**El tipo de navegador web que utilices es también elección tuya**. Pero es recomendable utilizar siempre las últimas versiones disponibles para evitar problemas de seguridad e incompatibilidades.

|  |  |
| --- | --- |
| **Para Windows** | Mozilla Firefox, Google Chrome, Safari, Opera, Internet Explorer, etc. |
| **Para Macintosh** | Mozilla Firefox, Safari, Google Chrome, Internet Explorer, etc. |
| **Para Linux** | Konqueror, Opera, etc. |

**Es recomendable disponer de 2 o 3 tipos de navegadores diferentes**, ya que así podrás probar la compatibilidad de tu página web y ver si tu código fuente de JavaScript se ejecuta correctamente en todos ellos.

Para terminar de configurar tu entorno de trabajo, **es necesario poder ejecutar el navegador y el editor web simultáneamente**, ya que el **flujo típico de trabajo en JavaScript será**:

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

Esa secuencia de grabar-cambiar-recargar la repetirás tanto que llegará a ser prácticamente un acto reflejo.

**Otro aspecto muy importante**: **la validación**. Puedes ahorrarte muchas horas de comprobaciones simplemente asegurándote de que tu código HTML es válido, ya que si tu código HTML contiene imperfecciones, es posible que tu JavaScript o CSS no funcionen de la manera esperada.

[The W3C Markup Validation Service](https://validator.w3.org/)

**Ofrece tres formas de introducción de tu código para la validación**:

* copiando y pegando tu código en un formulario
* enviando el fichero .html
* o bien indicando la dirección URL dónde se encuentra nuestra página web.

Para saber más sobre:

* [HTML 4.01 Specification (w3.org)](https://www.w3.org/TR/html4/)
* [HTML5 Standard (whatwg.org)](https://html.spec.whatwg.org/multipage/)
* [XHTML 1.0: The Extensible HyperText Markup Language (Second Edition) (w3.org)](https://www.w3.org/TR/xhtml1/)
* [Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification (w3.org)](https://www.w3.org/TR/CSS21/)
* [El Servicio de Validación de CSS del W3C](https://jigsaw.w3.org/css-validator/)

# 4.- Integración de código JavaScript con HTML.

Vamos a ver la forma de integrar el código de JavaScript en tu código HTML.

|  |  |
| --- | --- |
| **Formas** | **Cómo hacerlo** |
| **Integrado directamente en el código HTML** |  |
| **Enlazando a un archivo .js** | No escribir ningún código de JavaScript entre esas etiquetas cuando uses el atributo src. |
| Ésta sería **la forma más recomendable**, ya que así se consigue   * Una **separación entre el código** y la estructura de la página Web * **podrás compartir código entre diferentes páginas**. * **Conseguirás que las páginas carguen más rápido**, ya que el script solo se tendrá que cargar una sola vez para todas las páginas que lo utilicen. |

**Formas de referenciar el fichero origen .js de JavaScript**: dependerá de la localización física de ese fichero.

* Referencia absoluta al fichero: la ruta tendrá que empezar por http://.
* En otro caso, tendrás que poner la ruta relativa donde se encuentra tu fichero .js

**Ejemplos**:



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**. Nada de lo que tu navegador descargue para mostrar la página web podrá estar oculto a la vista de cualquier programador.

## 4.1.- Integración de código JavaScript con HTML (I).

* A veces te puedes encontrar que tu script se va a ejecutar en un navegador que no soporta JavaScript. ¿Qué se puede hacer? Indicar con la etiqueta **noscript** que te permitirá mostrar un texto indicando que ese navegador no soporta JavaScript:

<noscript>Texto informativo</noscript>

* Si estamos trabajando con **XHTML**, la **sintaxis** para insertar un **código de JavaScript**:

<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 son ampliamente utilizados en JavaScript. Por ello, **la solución es encapsular las instrucciones de JavaScript en la sección CDATA**:

<![CDATA[

XHTML permite cualquier tipo de carácter aquí incluyendo < y el símbolo &

]]>

Al ser más complejo de escribir código de JavaScript, **es en XHTML razón de más para integrar el código en un fichero externo**.

* **Cómo proteger el código en JavaScript del uso fraudulento** por otros programadore so visitantes a tu página:
  + **Incluyendo un mensaje de copyright** en tu código fuente (opción recomendada). Pero no evita que alguien copie tu código y elimine tu mensaje de copyright.
  + **Utilizar herramientas de ofuscación del código para hacerlo ilegible** al programador intruso que quiera utilizarlo, eliminando 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.
    - [Javascript Obfuscator - Protects JavaScript code from stealing and shrinks size - 100% Free](https://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.