Desarrollo Entorno Cliente

# 1.A Arquitecturas Y Lenguajes De Programación En Clientes Web.

## Introducción

La World Wide Web ( o simplemente Web) es un entorno heterogéneo que ha sufrido una evolución en su concepción, su usabilidad, tecnologías empleadas desde su creación en los comienzos de los años 90.

Concebida y creado por Tim Berners-Lee y su compañero Robert Cailliau del CERN en 1989. “Había una necesidad de una herramienta colaborativa que soportara el conocimiento científico”.

Crearon un prototipo basado en el concepto de hipertexto. Basados en unos protocolos y especificaciones que han sido adoptados universalmente e incorporados a Internet.

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 puede entender tres cosas distintas:

* El proyecto inicial del CERN
* El conjunto de protocolos desarrollados en dicho proyecto.
* El espacio de información formado por todos los servidores interconectados. (se hace referencia a este último concepto)

Muchas de las discusiones sobre Diseño Web o Desarrollo Web son confusas.

Componentes como diseño gráfico o programación, forman parte. La creación y organización de contenido (arquitectura de la información) consideran la parte más importante del Diseño Web. También la facilidad de uso, el valor y funcionalidad del sitio web, su accesibilidad, publicidad, etc.

El desarrollo Web ha sido y sigue estando muy influenciado por múltiples campos (campo multidisciplinar) como el de las nuevas tecnologías, los avances científicos, el diseño gráfico, la programación, las redes, el diseño de interfaces de usuario, la usubilidad y una variedad de múltiples recursos.

Tres fases de concepción:

* Web 1.0, estado original se utilizan documentos estáticos y formularios básicos, se utiliza para páginas personales o empresariales sólo para anunciar sus servicios.
* Web 2.0, web social donde se comparte información, interoperabilidad, el diseño centrado en el usuario y la colaboración.
* Web 3.0, web semántica, una red de datos que pueden ser procesados por máquinas, la inteligencia artificial es capaz de analizar estos datos.

Cinco áreas que cubren las facetas del Diseño Web:

* **Contenido**: incluye la forma, presentación y organización del contenido del sitio.
* **Visual**: hace referencia a la plantilla empleada en un sitio web, utilizando HTML, CSS, elementos gráficos y el aspecto visual es el más importante en el Diseño Web.
* **Tecnología**: tal como HTML, CSS pero también empleando técnicas/lenguajes de programación.
* **Distribución**: la velocidad y la 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 relacionada con algún aspecto 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, por ejemplo, una página personal generalmente no tiene las consideraciones económicas que tendría una web que va a vender productos en Internet.

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

* **Server side**: incluye el hardware y software del servidor Web así como diferentes elementos de programación y tecnologías incrustadas (CGI escritos en PERL, Java, base de datos)
* **Client side**: referente a navegadores web y está soportado por tecnologías como HTML, CSS, JavaScript, los cuales se utilizan para crear la presentación de la página o proporcionar características interactivas.
* **La red**: describe los diferentes elementos de conectividad utilizados para mostrar el sitio web al usuario.

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 desarrollador Web.

## 2.- Lenguajes de Programación en clientes web.

Dos grupos básicos: **client-side y server-side**

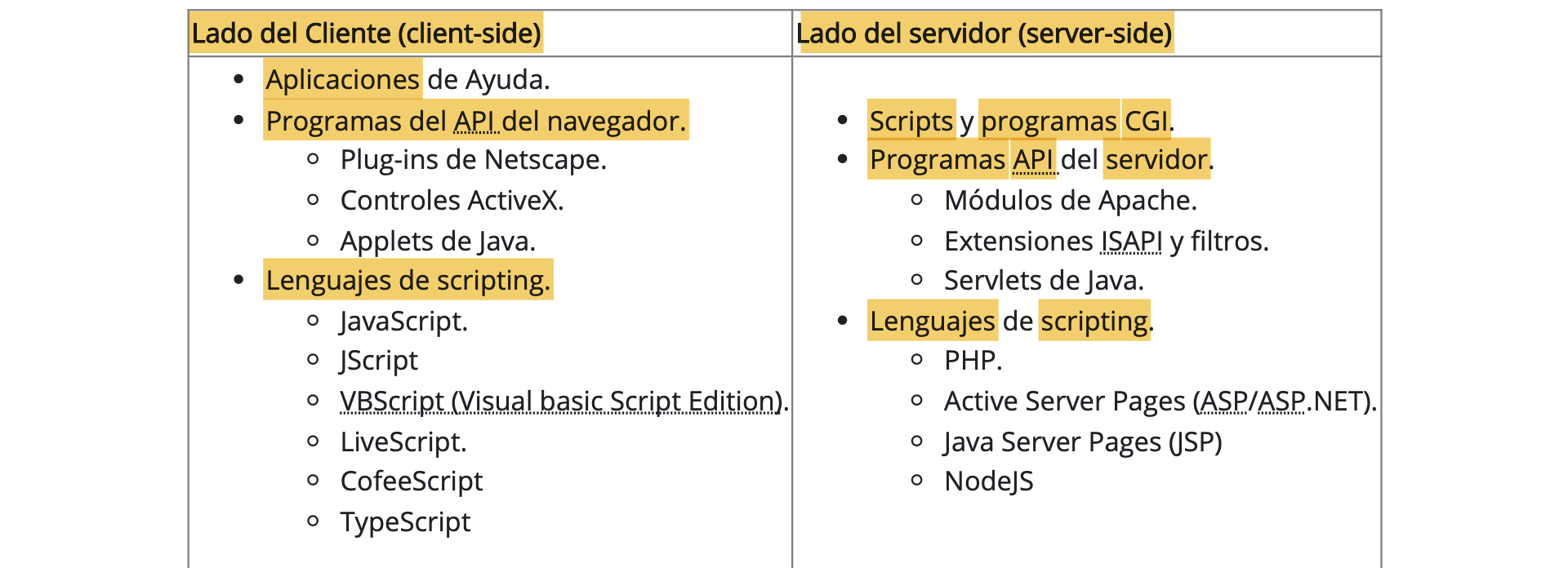
Las tecnologías Client-side son aquellas que son ejecutadas en el cliente y generalmente en un navegador web.

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. La realidad es que cada tecnología tiene sus pros y sus contras. Pero son complementarios unos con otras.

Dos variantes de lenguajes de programación en clientes web:

* 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)



Hemos escogido el lenguaje de programación en el lado cliente porque es el que está mayormente soportado por todas las plataformas: *JavaScript*.

**Cuatro** capas de desarrollo web:

* **Comportamiento** (JavaScript)
* **Presentación** (CSS)
* **Estructura** (DOM / estructura HTML)
* **Contenido** (texto, imágines, vídeo, etc)

## 2.1.- Características.

Los lenguajes de programación para clientes web no son un reemplazo de la programación en el lado de 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.

Razones:

* **Primero**: JavaScript por sí mismo no puede escribir ficheros en el servidor. Cede los datos al lenguaje del servidor encargado de la actualización de datos.
* **Segundo**: no todos los clientes web ejecutan JavaScript. Por incopatibilidad. Aun así nuestra página web debería ser completamente funcional con JavaScript desactivado.
* **Tercero**: integración servidor – cliente mendiante Ajax. El proceso asíncrono de AJAX se ejecuta en el navegador del cliente y emplea JavaScript. Este proceso se encarga de comunicarse con el servidor antes de enviar la petición mediante el envío de datos en formato XML, JSON o texto. Los datos devueltos por el servidor pueden ser examinados por JavaScript en el lado del cliente.

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, plu-ins o applets Java según la elección del usuario.
* **Pre-procesar** datos en el cliente antes de enviarlos al servidor.
* **Modificar** estilos y contenido en los navegadores de forma dinámica e instantáneamente.
* **Solicitar** ficheros del servidor y enviar solicitudes de lectura y escritura a los lenguajes de servidor.

JavaScript cumple con los estándares de ECMAScript (ES). Edición actual la ES6 o ES2015.

Existen otros lenguaje de scripting que permiten compilar a JavaScript y que aportan una sintaxis más reducida, compata u orientada a programación funcional que compila a JavaScript y que pueden ser también utilizados para escribir programas en el lado cliente. *TypeScript, CoffeScript, LiveScript, etc.*

Los lenguajes de script no se usan solamente en las páginas web. Los intérpretes de JavaScript están integrados en múltiples aplicaciones de uso cotidiano. Estas aplicaciones proporcionan su propio modelo de acceso y gestión de los módulos que componen la aplicación y para ello comparten el lenguaje JavaScript en cada aplicación. *Google Desktop gadgets, adobe acrobat, dreamweaver, openoffice.org, google drive, etc.*

Cada vez está más extendido en el desarrollo en el servidor proyectos como NodeJS, aplicaciones escritas en JavaScript aumentando notablemente la productividad al trabajar con un solo lenguaje en todas las capas de desarrollo.

## 2.2.- Compatibilidades

JavaScriptes Interpretado por el cliente. Existen múltiples clientes o navegadores que soportan JS (Firefox, Chrome, Safari, Opera, IE, Edge, 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 característica en cada uno de ellos.

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

A veces la incompatibilidades tienen como origen el código fuente HTML y no en sí el código JS. Por tanto, el código HTML es importante que siga las especificaciones del estándar W3C. Ej: Validador W3C.

Limitaciones en el uso de JavaScript:

* No todos los navegadores soportan lenguaje de script en el lado del cliente.
* Algunos dispositivos móviles tampoco podrán ejecutar JS.
* JavaScript no es totalmente compatible con los diferentes navegadores.
* La ejecución de código JS en el cliente podría ser desactivada por el usuario de forma manual.
* Algunos navegadores de voz, no interpretan el código de JS.

## 3.- Herramientas y utilidades de programación

Un aspecto importante es elegir adecuadamente que tipo de herramientas y utilidades adicionales se van a usara para programar con JS.

Decidir qué navegador o navegadores se van a usar para la comprobación y compatibilidad del código.

El editor de texto depende de ti pero no se recomienda editores del estilo WYSIWYG (Whay You See is Whay You Get).

Un factor importante al momento de elegir el editor es ver la facilidad con la que se pueden grabar los ficheros con extensión .html.

Características del editor muy favorables:

* Sintaxis con codificación de colores (resaltado de color) para los diferentes lenguajes.
* Verificación de sintaxis, que marque errores de código.
* Diferenciar los comentarios del resto de código.
* Generar automáticamente partes de código (snippets).
* Extensible añadiendo nuevas funcionalidades.
* Disponga de utilidades adicionales.

Editores gratuitos:

* Atom
* Brackets
* Notepad++
* Sublime text

Entorno desarrollo integrados:

* Aptana studio
* Eclipse
* Netbeans
* Visual studio code

Otro componente obligatorio es el navegador web y no es necesario tener conexión a Internet. Tipos de navegares:

* Chromiun
* Chrome
* Konqueror
* M Internet Explorer
* Mozilla Firefox
* Opera
* Safari

Se recomienda tener 2 o 3 tipos de navegadores diferentes, para comprobar la compatibilidad y el código JS.

Entorno de trabajo, se necesita el poder ejecutar tu editor web de forma simultánea con el navegador ya que el flujo típico de trabajo sera:

* Introducir HTML, JavaScript y Css en el documento original en el editor web.
* Guardarlo en disco.
* Cambiarte al navegador web.
* Realizar una de las siguientes 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. Grabar-cambiar-recagar.

Otro aspecto muy importante es la validación. Se ahorra 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 resultados obtendrás entre los diferentes tipos de navegadores.

W3C desarrollo un validador que te permitirá chequear si tu página web cumple las especificaciones indicada por el elemento DOCTYPE que se incluye al principio de cada página web.

<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.

## 4.- Integración de código JavaScript con HTML.

Distintas formas de ejecutar JS en nuestro navegador.

* Bookmarklest, y permiten ejecutar pequeñas aplicaciones JS sin necesidad de cargar un documento ya que la URL del bookmarklet se puede almacenar en los marcadores del navegador.
* Integrar el código JS en el código HTML. Los navegadores web te permiten varias opciones de inserción de código de JS.

Por ejemplo:



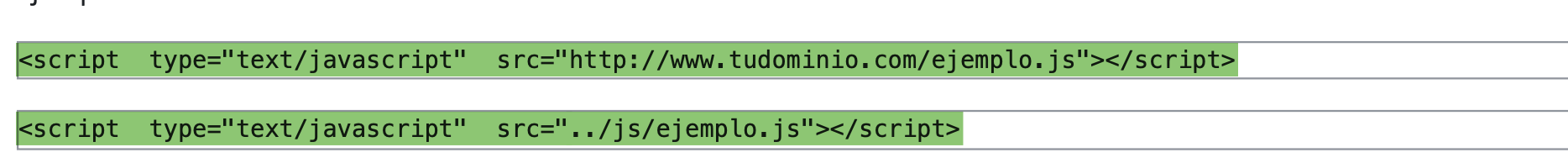
* Otra forma de integrar el código de JavaScript es incrustar un fichero externo que contenga el código de JS. Muy recomendable. Separación entre código y la estructura de la web, centralizar el código para depuración de errores, compartir código, claridad en desarrollo, modularidad, seguridad del código, mayor rapidez de carga de las páginas.

Por ejemplo:



Si necesitas cargar más de un fichero .js repite la misma instrucción cambiando el nombre del fichero. **Atención**: no escribas ningún código de JS entre esas etiquetas.

Para referenciar el fichero origen .js dependerá de la localización física de ese fichero. Se utilizará un referencia absoluta o relativa.

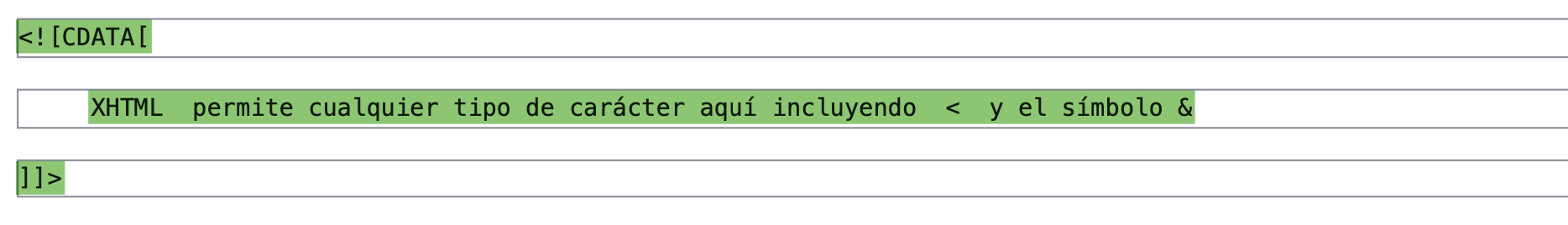


A veces te puedes encontrar que tu script se va a ejecutar en un navegador que no soporta JS. Disponemos de la siguiente etiqueta para indicar que ese navegador no soporta JS.

En XHTML:



O encapsular las instrucciones de JS con CDATA:



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

Proteger el código de JS es imposible hacerlo.

Para que el código de JS pueda ejecutarse correctamente deberá ser cargado por el navegador web y por lo tanto su código fuente deberá estar visible al navegador.

Puede nuestro código incluir un **copyright**.

Lo que se puede hacer es que el código sea más difícil de leer o incluso hacerlo ilegible para un humano (*ofuscar*). Se utiliza software para ofuscar el código, comprimiendola o haciéndola ilegible y esto lo hará más rápido.

<https://www.javascriptobfuscator.com/>

<https://github.com/mishoo/UglifyJS>

Lo mejor es cambiar el paradigma y piense de una manera diferente. En lugar de proteger tu código, lo mejor es promocionarlo y hacer ostentación de él. O liberando el código con 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, trabajos, etc.