

TEMA I

**INTRODUCCIÓN AL DESARROLLO WEB**  
*CLIENT-SIDE*

## ÍNDICE

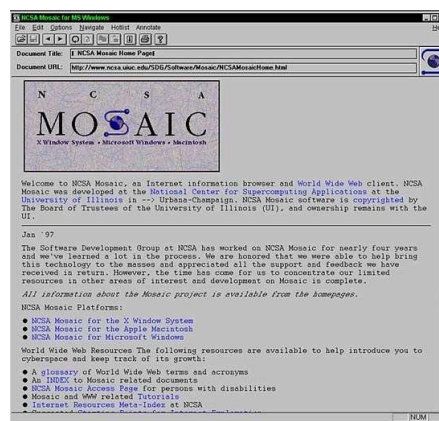
<b>I</b>	<b>DESARROLLO DE APLICACIONES WEB .....</b>	<b>3</b>
I.1	Áreas en el diseño web .....	3
I.2	¿Pero qué es una aplicación web?.....	4
I.3	Proceso de desarrollo de aplicaciones web .....	5
I.4	Front-end y Back-end .....	7
<b>II</b>	<b>LENGUAJES DE PROGRAMACIÓN EN CLIENTES WEB.....</b>	<b>10</b>
II.1	Orientado a la programación del lado cliente.....	11
II.2	Compatibilidades.....	13
II.3	Seguridad .....	14
<b>III</b>	<b>BIBLIOGRAFÍA CONSULTADA Y RECOMENDADA.....</b>	<b>16</b>

## I DESARROLLO DE APLICACIONES 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 de 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 la **popular interfaz Mosaic**, desarrollado por el NCSA (*National Center for Supercomputing Applications*), organismo norteamericano relacionado con la investigación en el campo de la Informática y las telecomunicaciones.



Todos los prototipos y desarrollos posteriores crecieron bajo la guía del consorcio **W3C**, una organización con base en el MIT de Massachusetts y que **se responsabiliza de desarrollar y mantener los estándares 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 de usuario, la usabilidad y una variedad de múltiples recursos. De este modo, podemos **entender el desarrollo Web como un campo multidisciplinar**.

### I.1 ÁREAS EN EL DISEÑO WEB

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 (cada vez más en desuso) y puede incluir elementos gráficos para decoración o para navegación.

- ✓ **Tecnología:** En este contexto hacemos referencia a los diferentes tipos de elementos interactivos de un sitio web, ya sea con un usuario o con una aplicación, contruidos empleando determinadas técnicas de programación. Tenemos dos tipos:
  - Tecnologías orientadas al cliente: Son aquellas que permiten **crear interfaces de usuario y establecer comunicación con el servidor** basadas en *HTML*, *CSS* y *JavaScript*, en este caso, el traductor (intérprete) suele ser el mismo navegador.
  - Tecnologías orientadas al servidor: **Permiten implementar comportamientos de la aplicación web en el servidor**, los lenguajes más utilizados son *Java EE*, *lenguajes .NET*, *PHP*, *Ruby on Rails*, *Python*, *Django*, *Groovy*, *Node.js*, etc...
- ✓ **Distribución:** la velocidad y fiabilidad con la que un sitio web se distribuye en Internet o en una red interna corporativa está relacionada 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. En este sentido una página personal generalmente no tiene las **consideraciones económicas** que tendría una web que va a vender productos en Internet.

Podríamos visualizar este punto en la imagen de la derecha:



## 1.2 ¿PERO QUÉ ES UNA APLICACIÓN WEB?

En informática, una **aplicación** es un software que permite al usuario realizar una **determinada tarea o servicio**. Las aplicaciones clásicas se crean en lenguajes como C o C++ que ponen a disposición del programador todas las capacidades de la computadora. Actualmente a este tipo de aplicaciones se las conoce como **aplicaciones de escritorio**.

Una **aplicación web** es, según Wikipedia, **aquella aplicación que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador**. De este modo, tenemos que los lenguajes en los que se puede programar estas aplicaciones son aquellos que un navegador es capaz de traducir o interpretar.

Hoy en día las aplicaciones web son el tipo de aplicaciones más popular ya que, independientemente del sistema, todos los usuarios tienen instalado en sus equipos algún

navegador (*Google Chrome, Internet Explorer, Mozilla Firefox...*) Los navegadores están presentes incluso en dispositivos más pequeños como tabletas o smartphones.

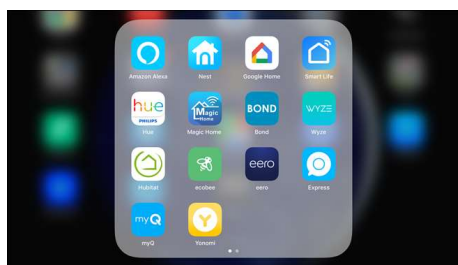
## 1.3 PROCESO DE DESARROLLO DE APLICACIONES WEB

### 1.3.1 No todo son páginas web

Hoy en día tenemos una gran cantidad de dispositivos conectados a la nube: el portátil, el teléfono móvil, la tableta, la televisión... y con la *Internet of Things* (IOT) estos dispositivos irán más en aumento. Una de las tareas habituales que hacemos con alguno de estos dispositivos es navegar por Internet visitando páginas web.



Existen otros dispositivos y aplicaciones que, aunque no sirvan para navegar por la web, sí que hacen consultas a servicios web. Imaginemos por ejemplo un aplicativo o aplicación web para controlar una casa domótica donde podemos poner la calefacción a distancia; o una aplicación para el móvil que gestione un calendario que se sincroniza continuamente y nos avisa de cuándo tenemos una reunión.



Así, tenemos que tomar conciencia de que lo que nos permite hoy por hoy la tecnología web, es **mucho más que visitar páginas web.**

### 1.3.2 Cliente-servidor

Si algo tienen en común todas estas aplicaciones web es que se basan en la arquitectura cliente-servidor donde una aplicación, que es el cliente web, **realiza peticiones (request) a un servidor web** que le envía (o no) los datos solicitados.

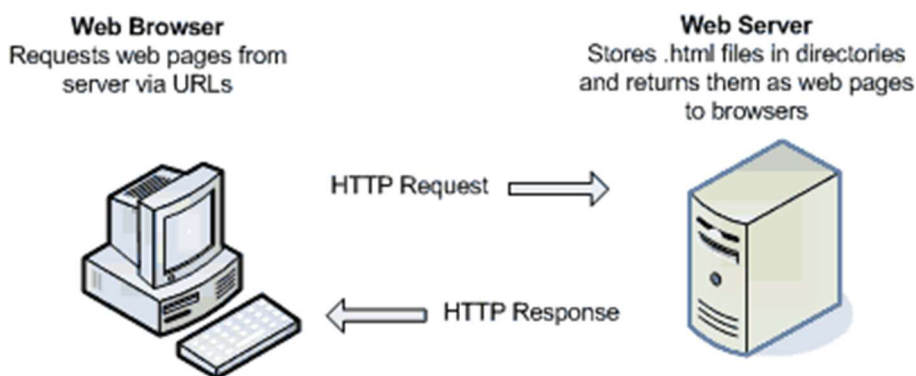
Cuando hablamos de **cliente web** podemos pensar en un usuario usando un navegador web, esto no es exacto: **el cliente de la conexión sería el browser** o navegador. Por otro lado, una aplicación que nos permita consultar el horario de un tren de cercanías también sería un cliente que pide a un servidor el horario actualizado. **Cualquier aplicación que solicite información a un servidor se podrá considerar un cliente web.**

En el otro lado tenemos los **servidores web**, que **sirven (response) el contenido web que tienen alojado**. Ejemplos de servidor web son el *Apache Web Server* o *Internet Information Server*, aunque también es bastante común los servidores de aplicaciones libres *JOnAS* de ObjectWeb y *Tomcat* de Apache, aunque este último está más orientado a *JavaEE* almacenando servlets (clases de Java utilizadas para ampliar las capacidades de un servidor) y código JSP.

## Funcionamiento en la web del modelo cliente-servidor

Supongamos que estamos con nuestro portátil, y consultamos una web de viajes desde el navegador Chrome. Veamos el camino que hacen los *paquetes IP*:

- ✓ **Paso1:** Supongamos que desde el navegador *Chrome*, en la barra de navegación, ponemos la URL del sitio web al que queremos llegar (por ejemplo, [www.visitmordorinautumn.com](http://www.visitmordorinautumn.com)) e intentamos acceder a ella.
- ✓ **Paso2:** Los servidores DNS localizan la dirección IP del servidor donde está alojado el sitio web. La información enviada se trocea en forma de paquetes IP (numerados para posibilitar su posterior ensamblaje) que viajarán por Internet atravesando diferentes capas de seguridad (routers, firewalls...) hasta alcanzar la entidad de esa IP.
- ✓ **Paso3:** Los paquetes IP finalmente se juntan en el destino, donde podríamos encontrarnos con un servidor web Apache que aloje el script *index.php*. Este script se ejecuta (se procesa la parte de PHP y el resultado se junta con la parte de HTML), y finalmente el resultado (el sitio web) se sirve al cliente haciendo el camino inverso.



- ✓ **Paso4:** Cuando la página web llega al destino, en el navegador web podemos visualizar la información solicitada. El navegador web, nuestro cliente, es el responsable de maquetar correctamente la página web a partir del contenido HTML, las hojas de estilo CSS y el código JavaScript que contiene.

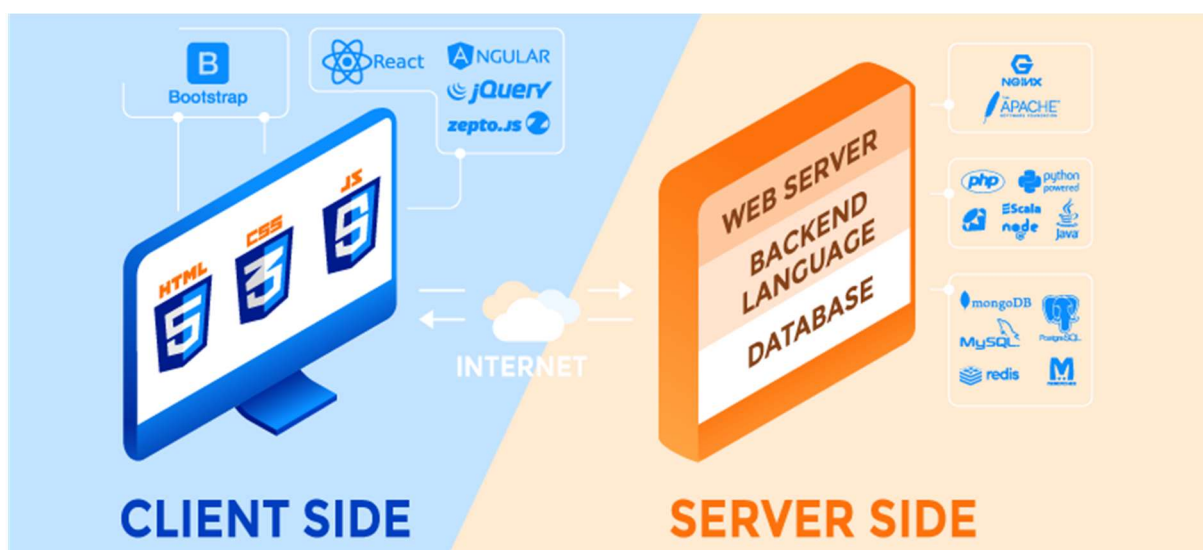
Podemos ver el código fuente que hemos recibido del servidor haciendo **Ctrl-u** en el navegador (fíjate que nunca verás en el código fuente el código PHP; por el contrario, sí que puedes ver el código *JavaScript*).

Es este entorno que rodea al cliente (browser) lo que estudiaremos en nuestro módulo, en contraposición con la parte de servidor (DWES). Tenemos que tener claro que del mismo modo que el código PHP se ejecuta en el servidor, el código JavaScript que contienen las webs se ejecuta en el cliente.

### 1.3.3 Server-side / Client-side / Red

Así tenemos tres partes identificadas en este proceso:

- **El lado del cliente** (*client-side*): este elemento hace referencia a los navegadores web y está soportado por tecnologías como HTML, CSS, lenguajes como JavaScript y controles ActiveX, los cuales se utilizan para crear la presentación de la página o proporcionar características interactivas. Es aquí donde tiene lugar la parte del software que interactúa con los usuarios o **front-end**.
- **El lado del servidor** (*server-side*): incluye el hardware y software del servidor Web así como diferentes elementos de programación y tecnologías incrustadas incluyendo tecnologías de servidor de bases de datos que soporten múltiples sitios web. Esta es la parte que procesa la entrada de datos desde el front-end, y que es conocida como **back-end**. Después de dicho procesamiento construirá y enviará una página de respuesta al cliente.
- **La red**: describe los diferentes elementos de conectividad (definida como la capacidad que tiene un dispositivo para poder conectarse a otros). Aquí se detallan los diferentes protocolos y material utilizado para poder realizar dicha conexión mostrando el sitio web al usuario.



## 1.4 FRONT-END Y BACK-END

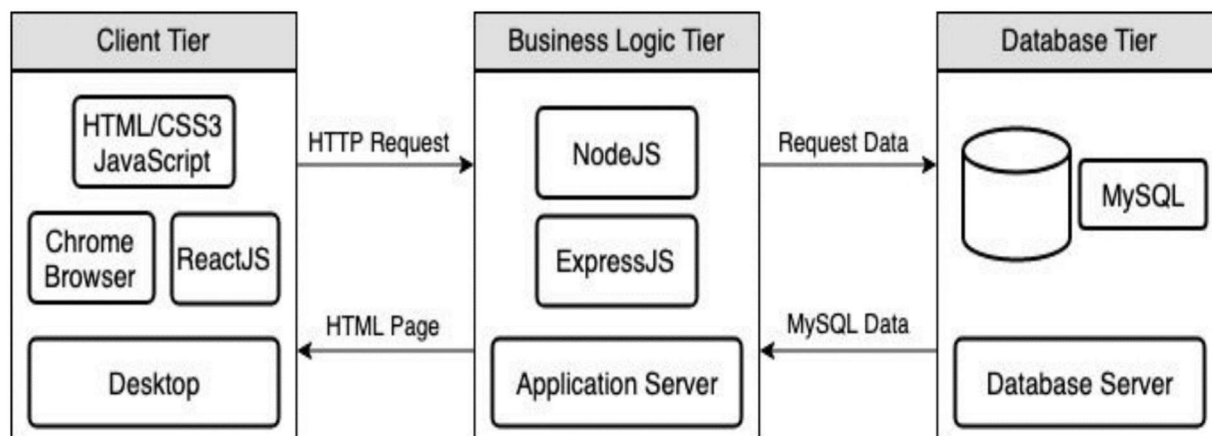
### 1.4.1 Arquitectura de tres capas

Las aplicaciones web actuales utilizan lo que se conoce como arquitectura de tres niveles o capas (*three-tier architecture*):

- **Capa de presentación** (de cliente o del navegador). Controla la parte de la aplicación web que se plasma en el navegador del usuario. Es decir, se encarga de la forma de presentar la información al usuario. Esta capa es la relacionada con la programación del lado del cliente o *front-end* programándose, como ya dijimos, en HTML, CSS y



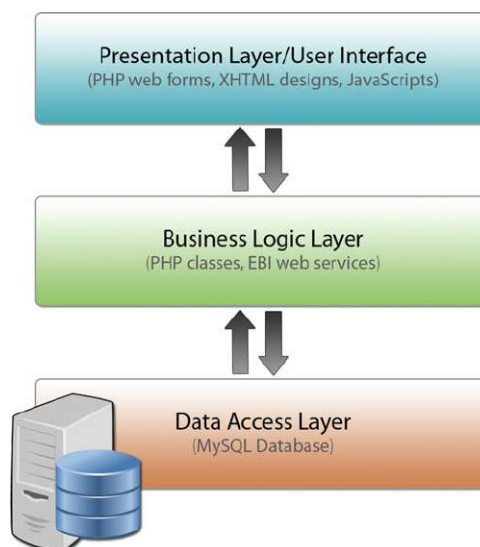
*JavaScript*. El código que procede de esta capa **se envía al servidor web**, esa comunicación se realiza por medio del protocolo http (más bien https).



- **Capa lógica** (de negocio, del servidor o de aplicación). Es la encargada de **gestionar el funcionamiento interno de la aplicación**. La capa se programa en lenguaje del lado del servidor como: *PHP*, *ASP*, *JavaScript* (hay JavaScript del lado del servidor) y otros. El código de esta capa se traduce por un *servidor de aplicaciones* que, normalmente, es un servidor web con módulos de traducción de esos lenguajes.

El servidor de aplicaciones traduce ese código, siendo su resultado otro código ya compatible con la **capa de presentación**. Por ejemplo, si hemos escrito esta capa usando el lenguaje *PHP*, el servidor de aplicaciones traducirá el código *PHP* produciendo código *HTML*, *CSS* y *JavaScript* que es el que se entregará al navegador. Al navegador no le llega el código *PHP* original.

- **Capa de base de datos** (o capa de datos). En esta capa **se almacena la información empresarial**, la cual va a seguir siempre un requerimiento básico: que su información **quede oculta a cualquier persona sin autorización**. Esta información suele ser administrada por un sistema gestor de base de datos (*SGBD*) cuya función es la de administrar la información de la empresa.



Es habitual encontrar otros tipos de servidores que sirvan para proporcionar recursos empresariales adicionales tales como vídeo, audio, certificados, correo, etc.

La capa lógica se comunica con esta capa para obtener datos que luego procesa y entrega, ya convertido en un formato interpretable, al navegador. De modo que el proceso de acceso a estos recursos queda oculto totalmente al navegador añadiendo una mayor seguridad al proceso.



### 1.4.2 Desarrolladores web (¿front-end, back-end o ...?)

El desarrollo web es muy diverso y, por ello, los trabajadores se suelen especializar solo en un aspecto de este. Existen *diseñadores web* en los que se valora la originalidad o el



buen gusto para elegir colores, formas y disposición de elementos en una página web, *programadores web* expertos en algún lenguaje del lado del cliente o servidor, *administradores de sistemas y bases de datos*, *arquitectos web*, etc.

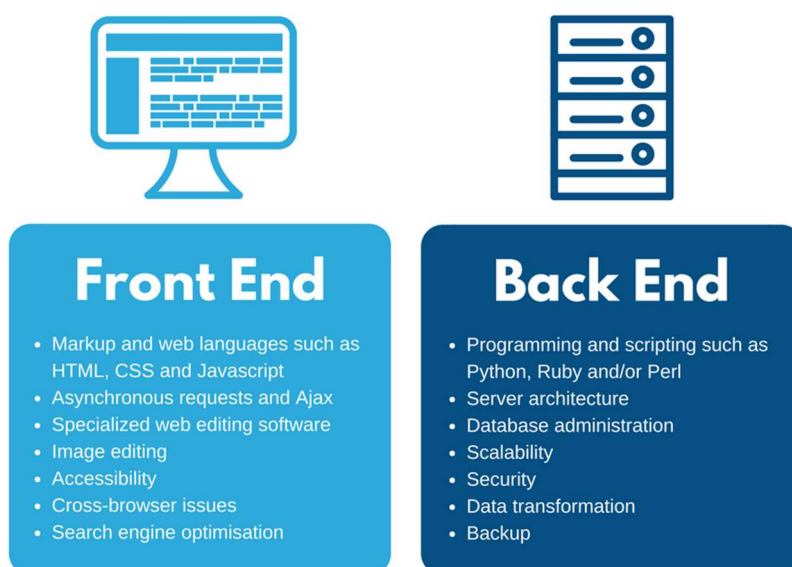
La separación del sistema en **front-end** y **back-end** es una abstracción que ayuda a mantener las diferentes partes del sistema cliente-servidor separadas, algo deseable en cualquier desarrollo software.

- **Back-end.** Parte no visible de la web donde encontramos las bases de datos o los scripts que se ejecutan en el servidor. Se corresponde con las capas lógica y de datos del modelo de tres capas. Los técnicos de back-end se encargan del todo el proceso de desarrollo software en el lado del servidor, como el acceso a la base de datos (*MySQL, MaríaDB, PostgreSQL, MongoDB, Oracle*, etc.), creación de servicios, etc. Sus técnicos programarán en lenguajes como *PHP, Ruby on Rails, Django, Node.js, .NET*, etc.

Sus objetivos son: un acceso rápido a los datos, una comunicación eficiente con la base de datos y el navegador, control de la seguridad, etc.

- **Front-end.** Parte visible de una web, como las hojas de estilo, el código HTML, los scripts que se ejecutan en el lado del cliente, etc.). Se corresponde con la capa de presentación. Aquí trabajan los diseñadores de la aplicación web y los programadores; los primeros están encargados de determinar lo que se conoce como *experiencia de usuario (UX)* así como el diseño visual de la aplicación, mientras que los segundos serán expertos en los lenguajes *HTML, CSS y JavaScript*.

Como objetivo de esta parte del desarrollo: optimizar la presentación y conseguir una interfaz final lo más agradable y funcional posible para los usuarios (*user-friendly interface*).



- **Full stack developer.** En las empresas desarrolladoras de aplicaciones web suelen haber técnicos especialistas en *back-end* y *front-end*. Un **Full Stack Developer** es un programador con un perfil técnico muy completo: sabe manejarse tanto en el *back-end* como en el *front-end*, sus diversos componentes así como con los diferentes sistemas operativos, sistemas de bases de datos y navegadores donde realizar y probar los desarrollos del software.

En definitiva, el *Full-Stack Developer* se adapta al perfil de un programador senior con experiencia.



Lo cierto es que la especialización en profesionales en la creación de aplicaciones web sigue siendo muy importante e incluso cada vez se especializa más. Así, sigue siendo normal, a la hora de optar a un puesto profesional de esta rama de la informática, que una oferta laboral se refiera a un **programador front-end** o a un **programador back-end**. Es más, en muchos casos el término *full-stack developer* se refiere a cualquiera de estos expertos con ciertos conocimientos extra en otras tecnologías, lo que le confiere un mayor dominio en conjunto del software a desarrollar.

Incluso, muchas veces, se requieren profesionales más específicos como diseñadores, administradores de bases de datos, especialistas en **DevOps**, desarrolladores de juegos, científicos de datos o especialistas en inteligencia artificial.

Por DevOps entendemos una práctica de la ingeniería del software que implica crear, probar e implementar el software en un entorno de producción.

## II LENGUAJES DE PROGRAMACIÓN EN CLIENTES WEB

Uno de los objetivos en la programación web es saber escoger la tecnología correcta para nuestro 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 (o front-end) y server-side (o back-end) 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, justo antes de enviar la información a la base de datos del servidor, **chequear el formulario en el lado del cliente** para asegurarnos que la información introducida es correcta. 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 con un **rápido acceso a la base de datos**.

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. Y es que *JavaScript* es el lenguaje de script más utilizado en la programación en el lado del cliente, y está soportado mayoritariamente por todas las plataformas o sistemas operativos.

Definimos *Lenguaje de Script* como aquél lenguaje de guiones o de órdenes que se almacena por lo general en archivos de texto plano y que será ejecutado por un programa intérprete que, en nuestro caso será el navegador.

Con todo esto, el desarrollo web del lado cliente se podría resumir los siguientes cuatro puntos:



---

## II.1 ORIENTADO A LA PROGRAMACIÓN DEL LADO CLIENTE

Los lenguajes de programación para clientes web no son un reemplazo de la programación en el lado del servidor. En el caso de JavaScript la razón principal es que este lenguaje, **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.

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.

### II.1.1 AJAX

Sin embargo, Uno de los caminos que más ha integrado la programación cliente con la programación servidor vino con el uso de AJAX (Asynchronous JavaScript And XML, gracias al objeto `XMLHttpRequest` creado por Microsoft para Internet Explorer 5.0).

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 código 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.



- La programación síncrona se ejecuta en secuencia, hasta que un proceso no termine el siguiente no comenzará.
- La programación asíncrona facilita que se pueden ejecutar varios procesos al mismo tiempo dando la sensación de ser más rápido.

### II.1.2 Entonces, ¿para qué usaremos JavaScript?

JavaScript está orientado a ofrecer las siguientes soluciones al desarrollador:

- ✓ 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, así como proporcionar una interfaz amigable para esos datos.
- ✓ Controlar múltiples ventanas o marcos de navegación, plug-ins, o applets de Java basados en las elecciones que ha hecho el usuario en el documento HTML.
- ✓ Preprocesar 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 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.

## II.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.

Para saber si una etiqueta, atributo, valor o palabra reservada es soportada por nuestro navegador o por otros podremos recurrir a la web <http://www.caniuse.com>

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. 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/>

Además, debemos 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, como ejemplo tenemos las 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 con accesibilidad por voz, no interpretan el código de JavaScript.



## II.3 SEGURIDAD

Lamentablemente, JavaScript proporciona un **gran potencial para diseñadores maliciosos que quieran distribuir sus scripts a través de la web**. Para evitar esto, los navegadores web cliente aplican dos tipos de restricciones:

- ✓ Por razones de seguridad cuando se ejecuta código JavaScript éste lo hace en un **espacio seguro de ejecución (sandbox)** en el cual 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 enviada desde otra web, tal como usuarios, contraseñas, o cookies**.

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".

### II.3.1 Limitaciones de JavaScript

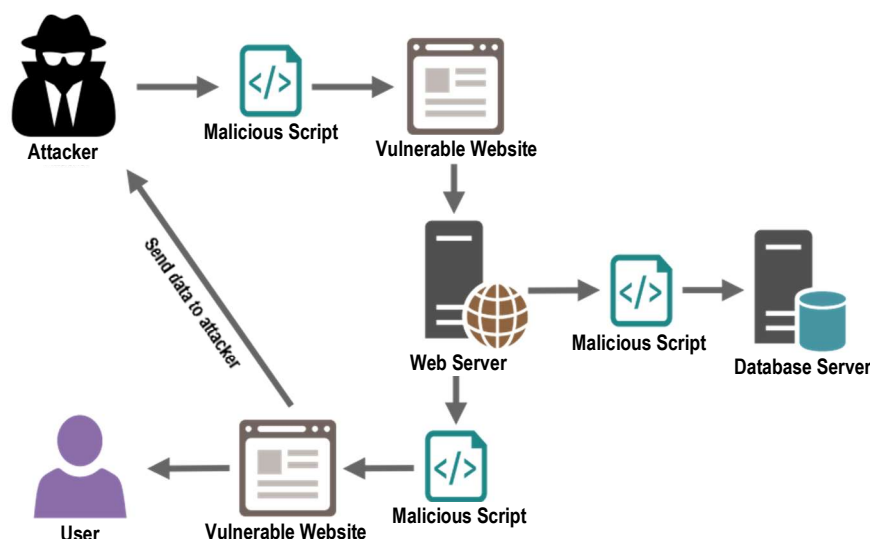
Es importante entender las limitaciones que tiene JavaScript y que, en parte, refuerzan sus capacidades de seguridad. JavaScript no podrá realizar ninguna de las siguientes tareas:

- Modificar o acceder a las preferencias del navegador del cliente, las características de apariencia de la ventana principal de navegación (cada vez menos permitido por los navegadores), capacidades de impresión, 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 *cross-site scripting* o **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 el usuario atacado. 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.

### II.3.2 Motor de ejecución

El motor de ejecución (o intérprete) de JavaScript es el encargado de ejecutar el código JavaScript en el navegador y por lo tanto es en él dónde recaerá el peso fuerte de la implementación de la seguridad. Entre otros, podemos 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* (aunque deberían).
- ✓ 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 las características que más resaltan y que permiten diferenciar a unos navegadores de otros son:

- ✓ La **rapidez** con la que sus motores de JavaScript pueden ejecutar las aplicaciones.

- ✓ La **seguridad y aislamiento** que ofrecen en la ejecución de las aplicaciones en diferentes ventanas o pestañas de navegación.

### III BIBLIOGRAFÍA CONSULTADA Y RECOMENDADA

-  *Desarrollo web en entorno cliente*. Juan Carlos Moreno Pérez, Edit. Síntesis
-  *Desarrollo web en entorno cliente con JavaScript*. Jorge Sánchez Asenjo, Edit. Garceta
-  <https://platzi.com/>
-  <https://developer.mozilla.org/es/docs/Web/JavaScript>
-  <https://www.w3schools.com/>
-  <http://es.wikipedia.org>