Temas Específicos para la preparación de la Oposición Cuerpo Superior de Estadísticos del Estado
Especialidad I de Estadística-Ciencia de Datos.

Bloque III. Almacenamiento y modelos de datos

Tema 8. Tecnologías web. II: lenguajes de scripts, CGI y cookies HTTP. Lenguajes de scripts: JavaScript, JScript y PHP. CGI: paso de información a un script CGI, ventajas y desventajas de CGI. Cookies HTTP.

AUTOR: María Teresa Avelino Carmona.

Asociación Profesional de Cuerpos Superiores de Sistemas y Tecnologías de la Información de las Administraciones Públicas

Creación: Junio 2021

ÍNDICE

1 Introducción

Las primeras páginas web (estamos hablando de finales de los 80 y principio de los 90) eran estáticas, escritas en HTML (que ya se ha visto en el tema anterior, el 7), con sólo imágenes y texto. No tenían contenido multimedia y eran raramente actualizables. Después, a finales de los 90, empezó a primar la vistosidad con el uso de multimedia pesada, primero con animaciones que usaban Flash Player (en su momento hizo furor pero no permite indexar sus contenidos y es una de las tecnologías ya obsoletas y que por defecto vienen deshabilitadas en los navegadores).

Otra solución que tuvo su momento de gloria en los años 90 fue el uso de marcos (frames), las páginas así construidas consistían en una barra lateral que contenía las diversas opciones del menú con una barra de scroll, y el contenido se ofrecía dentro de un rectángulo en mitad de la página web con otra barra de scroll para navegar dicho contenido. Hoy en día, las etiquetas <frame> y <frameset> ya no forman parte del estándar HTML más moderno, ya que se puede obtener la misma funcionalidad y mejor usando CSS.

Figura 1: Página web ejemplo de tecnología de marcos (usando wayback machine de web.archive.org)

La mejora en la apariencia gráfica de las web llegó de la mano de CSS (Cascading Style Sheets), "Hojas de estilo en cascada", que es un lenguaje de marcas (ver tema 7), enfocado a definir, crear y mejorar la presentación de un documento basado en HTML, al proveer de estilos y toda suerte de herramientas en el diseño de una web.

Después llegó la tecnología de **Applets de Java**, piezas de programación incrustadas en el código HTML de una web y que se ejecutan mediante un intérprete de comandos en el lado del navegador (Máquina Virtual de Java). Actualmente por motivos de seguridad, no es una tecnología muy en uso.

HTML v.5 permite el uso de gadgets y elementos de audio y vídeo directamente, tiene la ventaja de que se puede ejecutar también en navegadores de dispositivos móviles.

Sin embargo, algo que se usa a menudo y que confiere dinamismo a las páginas web son los lenguajes de scripting que, como veremos a continuación, se pueden implementar en el servidor o en el navegador del usuario que accede a la página web.

2 Lenguajes de scripts

Es a partir del surgimiento de las redes sociales, y las aplicaciones de negocio en la web (CRM, ERP, etc.) cuando empiezan a surgir las aplicaciones de scripting, para implementar la lógica de negocio y dar dinamismo a las webs.

Para imprimir ese dinamismo se puede hacer en el servidor, con un servidor web que implemente inteligencia con lenguajes tales como PHP, que genere las páginas web como respuesta a la información que se le presenta por parte del usuario. Pero también el dinamismo se puede implementar del lado de cliente, para lo que se requiere que su navegador implemente esa inteligencia capaz de interpretar las opciones del usuario mediante lenguajes JavaScript, con soporte para programación.

Los lenguajes de scripting representan en la actualidad aproximadamente un tercio de todos los lenguajes de programación más utilizados en el mundo. JavaScript lidera los lenguajes de scripting que se ejecutan en el lado del navegador del cliente, pero también los lenguajes del lado del servidor como PHP, Python, Ruby o Perl son lenguajes de scripting, en este caso del lado del servidor web.

Los lenguajes de programación convencionales son compilados. Esto es, primero se editan en un fichero llamado “código fuente”, donde se incluye la información de las librerías que se precisan utilizar para poder ejecutar dicho código, para seguidamente convertirse en código ejecutable mediante el proceso de compilación. Durante este proceso, si existen errores notables sintácticos, no se genera entonces el ejecutable.

El código binario ejecutable resultante está optimizado para ser ejecutado. Es por tanto, que los lenguajes compilados se utilizan para tareas que requieren alto rendimiento y el menor número de errores posibles.

A diferencia de los lenguajes compilados, los lenguajes de scripts son interpretados, lo que aumenta la carga en el procesador, pero descarga de tareas al programador. Con los lenguajes de scripting se puede programar software de manera más directa y con menos texto de código.

En muchos lenguajes de scripting es posible ejecutar los programas en modo interactivo, lo cual se denomina modo REPL (read-eval-print-loop), ciclo de lectura, ejecución y muestra. De este modo, si se produce algún fallo, el programador puede visualizar dónde está el fallo así como el contenido actual de las variables involucradas, lo que permite una buena depuración.

Otra de las características de los lenguajes de scripts es que tienen una menor complejidad en el uso de tipos de datos. Esto puede suponer una ventaja, aunque también un inconveniente si se desea elaborar programas complejos.

A continuación veremos algunas particularidades de los lenguajes de scripts tanto del lado del servidor como del lado de cliente.

2.1 Scripts del lado cliente

Un script del lado cliente es un programa incrustado en el código html del servidor web y que es interpretado y ejecutado por el navegador del usuario. Por tanto, estos scripts se ejecutan en el ordenador del usuario bien directamente (al abrir la página) o cuando se produce un determinado evento, como puede ser el pulsar sobre un enlace, mover el ratón por un área de la ventana de la web, cargar una imagen o modificar el tamaño de la ventana del navegador.

Estos scripts permiten crear páginas dinámicas y abordar tareas tales como la de modificar el comportamiento normal del navegador, validar formularios, realizar pequeños efectos visuales, etc. No obstante no podrán abordar tareas más complejas tales como el manejo de bases de datos.

El primer lenguaje usado para crear scripts fue el JavaScript de Netscape. Nacido con la versión 2.0 de este navegador y basado lejanamente en la sintaxis de Java, su utilidad y el casi absoluto monopolio que entonces ejercía Netscape en el mercado de navegadores permitieron que se popularizara y extendiera su uso.

Internet Explorer de Microsoft, comenzó a soportar este lenguaje en su versión 3.0, pero al mismo tiempo introdujo otro lenguaje con las mismas funciones: el VBScript, una derivación de BASIC. Pero este intento no llegó muy lejos, y el VBScript ha quedado para otras aplicaciones de Microsoft, como Access o Word.

2.2 Scripts del lado del servidor web

Como ya hemos visto, los scripts del lado del servidor web **sirven para realizar tareas más pesadas tales como extracción de datos de bases de datos**, actualización de transacciones y todo aquello que requiere de un cuidado y control porque puede comprometer la información y la lógica de negocio del sitio web en cuestión.

Una de las características de estos lenguajes de script es que, aunque el usuario accede a la operativa de la web a través de páginas HTML, el código fuente de los scripts permanece oculto para él. Otra característica de estos lenguajes es que es preciso contar con un canal de comunicación estable y síncrono entre el cliente y el servidor, a diferencia con los scripts del lado del cliente que se ejecutan de forma asíncrona en el lado del navegador del usuario.

Esta sincronicidad en las comunicaciones entre cliente y servidor supone una penalización en la capacidad de procesamiento de este último, que tiene que soportar tantos hilos o conexiones abiertas como peticiones de usuario esté sirviendo.

En los primeros tiempos de Internet, la programación del lado del servidor se reducía al uso de lenguajes más convencionales como C, Perl o lenguajes de comandos (como PowerShell o directamente el CMD, esto es, la línea de comandos del sistema), que **se comunicaban con el navegador del cliente para servirle la respuesta a la petición que había hecho mediante un interfaz de comunicación común** (Common Gateway Interface, CGI), que veremos más adelante en este tema.

Hoy en día, **muchos servidores web pueden ejecutar directamente los scripts sin necesidad del uso de CGI**. El lenguaje de programación del lado servidor más utilizado en la actualidad de este tipo es PHP, que veremos también un poco más adelante.

3 JavaScript, JScript, ECMAScript y otros lenguajes

La normalización en el uso de scripting en navegadores web se la debemos a Brendan Eich, quien desarrolló una especificación que llamó ***ECMAScript***.

Netscape fue el pionero en los navegadores web y durante un tiempo marcó el paso a la hora de implementar el desarrollo web, empleando dicha especificación primeramente en su navegador que lo llamó ***"Mocha",*** luego evolucionó a *LiveScript* y posteriormente acabó en JavaScript como resultado de una alianza entre Sun Microsystems y Netscape.

JSscript fue una adaptación de JavaScript para Internet Explorer, navegador líder durante mucho tiempo al ser monopolio de Windows.

3.1 ECMAScript

Netscape envió JavaScript a Ecma International (asociación internacional de estándares en Internet) para su estandarización y para que trabajasen sobre su especificación **ECMA-262**, que comenzó en noviembre de 1996.

Esta primera versión ECMA-262 fue aprobada por Ecma en 1997. Desde entonces ECMAScript ha ido evolucionando. Se puede decir que este estándar fue un compromiso entre Netscape inicialmente y Microsoft por estandarizar el lenguaje de scripting en navegadores de modo que fuera compatible entre unos y otros.

La versión actual es la 11, del año 2020, se puede consultar en el sitio web de Ecma:

A continuación se indican los rasgos principales de este lenguaje:

- **ECMAScript no es un lenguaje orientado a objetos como puede serlo Java**. Pero sí ofrece opciones para definir y utilizar clases. De este modo, se puede decir que es "basado en" pero "orientado a" objetos.
- Objetos predefinidos en el lenguaje que se pueden utilizar:
    - A nivel de control de ejecución (runtime): ***Object, Function, Boolean, Symbol*** y también varios de tipo ***Error.***
    - Para manipular valores numéricos: ***Math, Number, Date.***
    - Para manipular texto: ***String, RegExp.***
    - Para manipular Arrays: Hay nueve tipos de ***Array.***
    - Para manipular colecciones, conjuntos, etc. existen los objetos **Map** y **Set.**
    - Para datos estructurados ***JSON, ArrayBuffer, SharedArrayBuffer*** y ***DataView.***
    - Para manejar datos abstractos incluyendo funciones generadores y objetos: ***Promise.***
    - Para manejar objetos reflejados tenemos ***Proxy*** y ***Reflect.***
- Operadores: Se pueden usar de forma predefinida operadores unarios, multiplicativos, aditivos, relacionales, lógicos, binarios de todo tipo, operadores de asignación y también de coma flotante.
- Los scripts se pueden dividir en módulos si son muy largos.
- La sintaxis para escribir el script es muy similar a java (sólo que limitado, como ya se ha indicado).

3.2 JavaScript

Es importante destacar que, contra lo que se pudiera pensar y a diferencia del lenguaje de programación Java, JavaScript no es orientado a objetos, sino basado en objetos.

Por tanto, usa un modelo de instanciación de objetos muy simple que no precisa de conocimientos típicos de programación orientada a objetos tales como herencia y jerarquías de clases.

Los tipos de datos que se usan son muy simples: Numéricos, Booleanos y de tipo cadena de literales, String. Además, no es necesario declarar los tipos de las variables.

Al ser un lenguaje de scripting no necesita ser compilado para poder ser ejecutado ya que es interpretado. Esto comporta que los objetos se enlazan de forma dinámica: Es decir, los objetos deben existir en tiempo de ejecución.

Como ya se ha indicado anteriormente en ECMAScript, JavaScript comparte muchos rasgos con aquél, al ser el primero un estándar de referencia para este tipo de scripting.

Como casi siempre en programación, haremos un ejemplo “Hola Mundo” usando este lenguaje:

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

holamundo.html

```
<html>
<head>
<script languqge="JavaScript">
<!---
function Saludar() {
    alert("¡Hola, mundo!");
  }
// --->
</script>
</head>
<body>
<form>
<input type="button" name="Boton" value="Pulsame" onClick="Saludar()">
</form>
</body>
</html>
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

¡Hola, mundo!

Figura 2: Programa Hola Mundo. Elaboración propia a partir del script

El código JavaSript como vemos, se embebe en el HTML correspondiente haciendo uso de la etiqueta <script> donde indicamos que el lenguaje del mismo es JavaScript.

A continuación hacemos uso de las herramientas de scripting de dicho lenguaje:

- **Funciones:** Son las piezas fundamentales del lenguaje. Representa las tareas que deben realizarse y consta de las siguientes partes:
  - Nombre (de la función).
  - Parámetros (separados por comas y entre paréntesis).

- Inicio y fin delimitados por llaves: { }.

- **Variables: Se definen asignándoles bien un nombre o bien un valor.**
  - var factorial=6 ;
  - var factorial ;

En el ejemplo anterior la función la hemos llamado Saludar y no hemos incluido parámetros, pero podríamos definir una variable saludo de tipo *string* (var saludo;) y habérsela pasado como parámetro para que en el mensaje *"alert"* se expusiera dicho contenido.

Veremos a continuación algunos ejemplos un poco más elaborados:

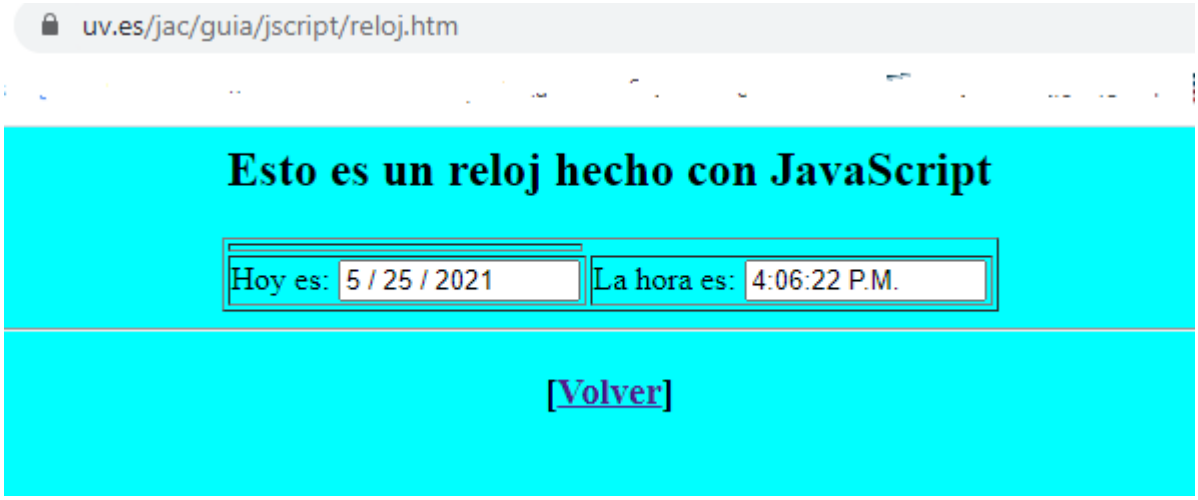


Figura 3: Programa Reloj. Fuente uv.es (tomado a su vez de scripts demo de Netscape)

Aquí en el script, remarcamos lo más interesante desde el punto de vista del aprendizaje:

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

```
<HTML>
<HEAD>

<TITLE>JavaScript Index</TITLE>

<script type="text/javascript" src="https://gc.kis.v2.scr.kaspersky-labs.com/A771B421-D021-884A-81CA-3A7DC7CF5218/main.js" charset="UTF-8"></script><script Language="JavaScript">

<!-- Helpers for JSI page...

// Navigation - Start

function goback(){

alert("Good Bye!");

history.go(-1);

}
```

```
function gettheDate() {
Todays = new Date();
TheDate = "" + (Todays.getMonth()+ 1) +" / "+ Todays.getDate() + " / " + (Todays.getYear() + 1900)
document.clock.thedate.value = TheDate;
}
// Navigation - Stop
// Netscapes Clock - Start
// this code was taken from Netscapes JavaScript documentation at
// www.netscape.com on Jan.25.96
var timerID = null;
var timerRunning = false;
function stopclock (){
        if(timerRunning)
                clearTimeout(timerID);
        timerRunning = false;
}
function startclock () {
        // Make sure the clock is stopped
        stopclock();
        gettheDate()
        showtime();
}
function showtime () {
        var now = new Date();
        var hours = now.getHours();
```

```
        var minutes = now.getMinutes();
        var seconds = now.getSeconds()
        var timeValue = "" + ((hours >12) ? hours -12 :hours)
        timeValue += ((minutes < 10) ? ":0" : ":") + minutes
        timeValue += ((seconds < 10) ? ":0" : ":") + seconds
        timeValue += (hours >= 12) ? " P.M." : " A.M."
        document.clock.face.value = timeValue;
        // you could replace the above with this
        // and have a clock on the status bar:
        // window.status = timeValue;
        timerID = setTimeout("showtime()",1000);
        timerRunning = true;
}
// Netscapes Clock - Stop
// end Helpers -->
</script>
</HEAD>
<BODY bgcolor="#00FFFF" onLoad="startclock()">
<CENTER>
<h2>Esto es un reloj hecho con JavaScript</h2>
<table border>
<tr>
  <td><form name="clock" onSubmit="0"></td>
</tr>
<tr>
  <td colspan=2>Hoy es: <input type="text" name="thedate" size=12
```

```
        value=""></td>

   <td colspan=2>La hora es: <input type="text" name="face" size=12

        value=""></td></form>

</tr>

</table>

</CENTER>

<hr>
<center>
<h3>
[<a href="javascr.htm">Volver</a>]
</h3>
</center>

</BODY>

</HTML>
```

**********+

Analizando el script, vemos que hay llamadas a funciones tanto estandarizadas como ***alert() o setTimeout()*** como definidas adhoc para el ejemplo. Igualmente, se incluyen objetos y variables en función de objetos predefinidos en el lenguaje, tales como la variable **seconds,** definida y asignada directamente como: ***"var seconds = now.getSeconds();".*** Hay que indicar también que se hace uso de formularios para devolver los resultados (*form name="clock" onSubmit="0"*).

Para finalizar, hay que indicar también que este lenguaje de scripting también se usa en aplicaciones que no tienen que ver con Internet, como por ejemplo en documentos PDF o en aplicaciones de escritorio para definir widgets (gadgets para mostrar utilidades tales como relojes y cronómetros, notas de texto, calculadoras, calendarios o información del tiempo).

3.3 JScript

Como vimos en la introducción, Microsoft con Internet Explorer deseaba apuntarse al carro de la compatibilidad con ECMAScript pero añadiendo sus propias funcionalidades e integración con la programación en Windows, por lo que desarrolló JScript.

JScript se incluyó por primera vez en Internet Explorer 3.0 y la última versión que lo incluye en la versión 9.

Las diferencias con JavaScript están por tanto en ese uso de librerías y rasgos típicos de la programación sobre Windows de finales de los 90 y comienzos del 2000: La tecnología ActiveX, componentes COM y acceso a datos a través de ADO.

De este modo, los scripts se implementan dentro de un motor de Active. Se pueden embeber en aplicaciones que usan OLE y Active Scripting tales como Internet Explorer o ASP (del lado del servidor).

Desde Windows 10 se puede deshabilitar y se desaconseja su uso en la actualidad por los problemas de seguridad que comporta, al hacer uso de librerías y tecnología de programación muy antigua.

En la actualidad ha sido reemplazado por JScript.NET, pero éste no es un lenguaje de scripting ya que precisa ser compilado.

Otra característica que deja abierto el estándar de ECMAScript y que difiere en JScript de JavaScript es en el uso de las APIs DOM. DOM (Documento Object Model) es un modelo de objetos (estándar definido por la W3C, World Wide Web Consortium) para la representación de documentos HTML, XHTML y XML.

Las instrucciones de JScript están estructuradas por líneas con uno o más elementos y símbolos del lenguaje y terminan en punto y coma (;).

Los bloques son conjuntos de líneas que están delimitadas por las llaves ({}). Los comentarios se pueden poner al final de la línea con las líneas (//)

A continuación un ejemplo muy sencillo:

```
function convert(pulgadas) {
pies= pulgadas / 12; // Esta es la primera instrucción
millas = pies / 5280;
millasNauticas = pies / 6080;
cm = pulgadas * 2.54;
metros = pulgadas / 39.37;
}
```

Con este script se puede convertir un valor dado en formato de longitud anglosajón al formato europeo.

3.4 TypeScript

Desde el año 2012 Microsoft lanzó una ambiciosa propuesta para poder simplificar JavaScript sin necesidad de pasar por la rigidez o pesadez de JScript o JScript.net. Desarrolló así un lenguaje de scripting que se puede considerar una ampliación de JavaScript con ampliación de tipos y objetos, así como extensiones para programas (Node.js y Deno.js). De este modo, **TypeScript es compatible con JavaScript,** pero en el caso de aquél, está pensado para **simplificar el trabajo engorroso cuando se abordan grandes proyectos usando JavaScript**, ya que incluye un compilador, que analiza el código fuente y lo traduce a JavaScript. Además, TypeScript está licenciado como software libre (licencia Apache 2).

Otra característica interesante de este lenguaje es que contiene ficheros de definición de tipos sobre las librerías de JavaScript, como se suele hacer en lenguajes como C ó C++, que describen la estructura de los ficheros de objetos disponibles. De este modo, otros programas pueden usar como si fueran entidades estas definiciones. Existen cabeceras para librerías de productos y bases de datos conocidas tales como jQuery, MongoDB y D3.js, y los módulos básicos de Node.js. que ya hemos comentado.

En el framework Microsoft Visual Studio 2013 Update 2 y posteriores, junto a C# y otros lenguajes de Microsoft se incluye también TypeScript. Visual Studio 2012 también puede soportar también TypeScript mediante la inclusión de una extensión a tal efecto.

Como curiosidad, **además de los tipos de JavaScript bien conocidos de "String" y "Number"** admite los siguientes tipos que define como básicos: Boolean, Array, Tuple (array de longitud fija), Enum (enumeración de elementos), Any (para trabajar con librerías externas), Void (típico de lenguajes como C), Never (para recoger errores es algo similar al "*exception*" de Java).

En TypeScript las variables se definen indicando de qué tipo de dato se trata.

Por ejemplo:

// Dato de tipo string
var name: string = 'Nombre';
name = 'Pepe'; // Es correcto
name = 2 // Es incorrecto

3.5 VBScript

Derivado de Visual Basic, pero adaptado a su uso en la web, comparte con aquél parte de la sintaxis, estructuras de control y funciones del lenguaje.

Su uso como lenguaje de scripting del lado cliente decayó en favor de JavaScript y de hecho, Internet Explorer lo incorporó en detrimento de VBScript pero éste se sigue usando en scripts del lado servidor en las páginas ASP (Active Server Pages).

Mientras que JavaScript es compatible con cualquier navegador, VBScript sólo con Internet Explorer.

Ejemplos:

<script type="text/vbscript">

<input type="button" value="Mensaje"
onclick="vbscript:MsgBox 'Enviar un mensaje al usuario', 32, 'PepePC'";>

</script>

Como deriva de Visual Basic, algunas funcionalidades como los cuadros de mensajes, en vez de usar la función *alert()* como en JavaScript vemos que se usan los típicos *MsgBox* de Visual Basic.

4 Lenguajes de Scripts en el Servidor: PHP

PHP fue escrito originalmente en C (su sintaxis actual sigue siendo muy parecida a este lenguaje).

El primer desarrollador fue Rasmus Leerdorf en 1994, que quería escribir una serie de utilidades para monitorizar su web personal, de ahí el acrónimo inicial "Personal Home Page".

Un año después, Leerdorf combinó este embrionario PHP con su intérprete de formularios, denominándolo *PHP/FI*, pero conocido más generalmente como **PHP 2.0**.

Al sumarse posteriormente al proyecto los desarrolladores Surasky y Gutmans, cambiaron el nombre del acrónimo por *"PHP: HyperText Preprocessor"* que, como curiosidad, es una definición recursiva, ya que incluye el propio acrónimo a definir.

Posteriormente en el año 2000 se añadió un motor de preprocesamiento conocido como "*motor Zend"*. Luego se han ido sumando numerosas utilidades hasta llegar a la versión actual, la versión 8.0.6, de mayo de 2021 (para ver más información y la última versión se recomienda consultar "**https://php.net", página web oficial de PHP.**

El gran éxito de PHP radica en su versatilidad de plataforma, puesto que permite generar páginas web tanto sobre servidores Linux como Windows. Asimismo, PHP se integra tanto con bases de datos como MySQL como con SQL Server, y con servidores web Apache como con IIS.

Como lenguaje de scripting del lado del servidor, el código que se genera con PHP suele ser procesado en un servidor web mediante un intérprete PHP. También se puede interpretar y ejecutar un código PHP cualquiera a través de una interfaz de línea de comandos (CLI).

Este intérprete puede ser un módulo instalado en el servidor, un demonio corriendo en dicho servidor, o bien puede ser recogido por un CGI. Una vez este código es interpretado y ejecutado, formaría parte de una respuesta de tipo HTTP.

Existen diversas plantillas, sistemas de gestión de contenidos y frameworks de desarrollo que sirven para organizar o facilitar la generación de esa respuesta HTTP.

Fuera del uso de Internet, PHP puede utilizarse también para aplicaciones gráficas autónomas o para el control de drones incluso.

Como curiosidad y respecto al tema de su licenciamiento como software es preciso indicar que **es una licencia Opensource (código abierto) pero no copyleft**, lo cual quiere decir que no se pueden modificar versiones anteriores y sacarlas al mercado libremente sin permiso del autor de la versión anterior.

A continuación veremos sus características principales y cómo se utiliza este lenguaje.

4.1 Características principales

Usando PHP se puede hacer todo lo siguiente:

- Generar páginas de contenido dinámico (CSS, JavaScript, etc.) que luego se ejecutará en el navegador de la parte cliente.
- Generar código HTML, imágenes, videos, PDF, XML, etc.
- Manipular ficheros en la parte del servidor: Crear, abrir, leer, escribir.
- Enlazar y utilizar bases de datos.
- Manejar los datos de los formularios web.
- Enviar y recibir cookies.
- Controlar el acceso de los usuarios al sitio web.
- Cifrar (encriptar) los datos de comunicación entre cliente y servidor.
- Usar extensiones (módulos).

- Permite programar usando el patrón MVC (Modelo-Vista-Controlador). Con este modelo se separa la parte de tratamiento y acceso a datos de la lógica de control del programa y todos ellos de la interfaz de usuario, manejando tres componentes independientes de esta forma.

Su uso se ha extendido enormemente en aplicaciones web para gestores de contenido (CMS que proviene del inglés Content Management Server). Por ejemplo WordPress 5.6 utiliza como base un servidor PHP 7.4 o superior. TYPO3 utiliza PHP 7.2 o superior. Drupal y Pico son otros dos CMS que están basados en PHP.

Mediante la herramienta Composer de PHP, se pueden añadir funcionalidades mediante librerías añadidas con Composer, que accede al repositorio Packagist (https://packagist.org)

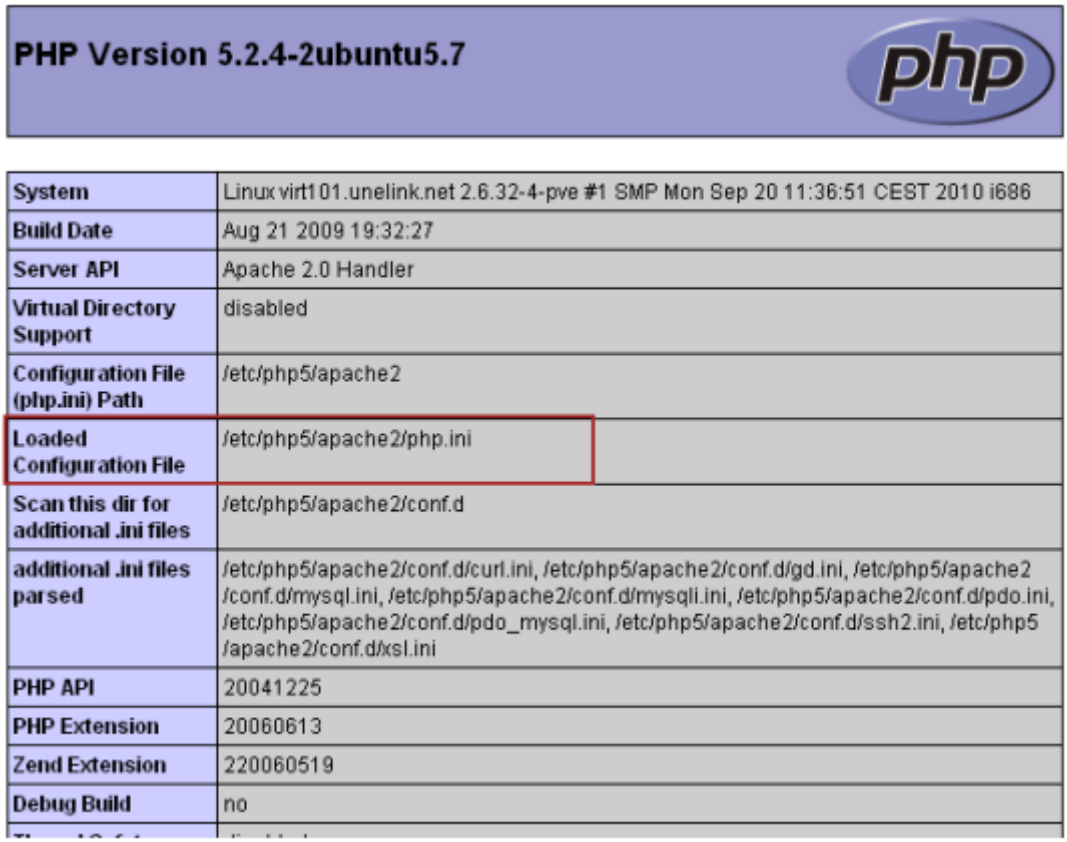
4.2 Instalación y módulos de PHP

Para instalar PHP se pueden seguir las Instrucciones de instalación de PHP en su sitio web oficial:

http://php.net/manual/en/install.php

La instalación de un servidor PHP tanto si es en Linux como en Windows requiere la instalación de una base de datos, bien sea MySQL o PostgreSQL que son típicas de Linux como SQLServer para Windows.

Para configurar PHP hay que tener en cuenta que el fichero de configuración se llama php.ini y se puede visualizar su ubicación y particularidades con el comando de información del a instalación de PHP: phpinfo()



PHP Version 5.2.4-2ubuntu5.7

System	Linux virt101.unelink.net 2.6.32-4-pve #1 SMP Mon Sep 20 11:36:51 CEST 2010 i686
Build Date	Aug 21 2009 19:32:27
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
additional .ini files parsed	/etc/php5/apache2/conf.d/curl.ini, /etc/php5/apache2/conf.d/gd.ini, /etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini, /etc/php5/apache2/conf.d/pdo_mysql.ini, /etc/php5/apache2/conf.d/ssh2.ini, /etc/php5/apache2/conf.d/xsl.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no

Figura 4: Configuración de PHP visualizada con phpinfo. Fuente: https://pressroom.hostalia.com/

Aunque se puede editar el texto de un script PHP con un simple bloc de notas (cambiando la extensión por defecto txt por la de php), existen editores preparados para editar estos scripts, como por ejemplo:

http://free-php-editor.com/

Como veremos con un par de ejemplos sencillos, para probar los scripts del lado del servidor, es preciso contar con tres componentes fundamentales: El analizador de PHP (que puede estar en el servidor o utilizarse un módulo CGI), un servidor web (como puede ser Apache o IIS) y un navegador web para analizar el resultado.

No obstante, también se puede crear un script PHP y ejecutarlo directamente desde la línea de comandos. Este tipo de scripts van más enfocados a tareas en background o accesos a bases de datos y se pueden programar con el cron de Linux o el planificador de tareas de Windows.

PHP-GTK es una extensión de PHP, que no está disponible en la distribución estándar de PHP pero que se puede descargar de http://gtk.php.net/ y que proporciona soporte para aplicaciones de escritorio que usen interfaces de ***GTK+.***

Figura 5: Ejemplo de uso de PHP-GTK

Para poder trabajar con librerías gráficas en la instalación sobre Windows se debe instalar el entorno de desarrollo **Visual Studio environment.**

Otra característica interesante de mencionar es que PHP se puede instalar en los Cloud (Nubes) de Microsoft Azzure y Amazon EC2.

Además de conectarse con base de datos o interfaces gráficas, PHP puede comunicarse con otros servicios estándar tales como LDAP, IMAP, NNTP, SMTP, POP3, HTTP o COM (en Windows). Con otros lenguajes de programación web se puede intercambiar datos usando WDDX. También puede utilizar objetos del lenguaje Java y reconvertirlos como objetos de PHP.

4.3 Ejemplos sencillos

En el manual de PHP (ver bibliografía) podemos encontrar gran variedad de ejemplos que nos pueden servir de base para construir nuestros programas.

```
<html>
 <head>
  <title>Prueba de PHP</title>
 </head>
 <body>
 <?php echo '<p>Hola Mundo</p>'; ?>
 </body>
```

```
</html>
```

Como de costumbre nuestro **script Hola Mundo**.

Para poderlo ver no podríamos abrirlo directamente invocando al navegador sino que tendríamos que hacer la llamada al servidor de PHP en nuestro host local.

Se podría ver usando entonces la llamada: ***http://localhost/hola.php***

El código que recibiría el navegador web y que interpretaría entonces sería:

```
<html>
 <head>
  <title>Prueba de PHP</title>
 </head>
 <body>
 <p>Hola mundo</p>
 </body>
</html>
```

Algo más complicado podría ser un formulario web, aquí además tenemos que utilizar los métodos de validación http (GET, POST). Con el método GET, los datos que se envían al servidor se ven en la misma dirección URL que se envía para recogerlos. El método POST inserta los parámetros en la solicitud HTTP para que el servidor web los recoja, de modo que el usuario no puede visualizarlos, lo que abunda en su seguridad.

En el siguiente ejemplo, vemos que se utiliza el método POST, y que los parámetros que se le envían son el nombre y la edad. Vemos que las variables que lo recogen son de tipo “text”, esto es, texto, aun cuando la edad se podría haber definido como de tipo numérico. Sin embargo, el propósito de este ejemplo no es hacer cálculos sino presentar información requerida al usuario.

```
<form action="accion.php" method="post">
 <p>Su nombre: <input type="text" name="nombre" /></p>
 <p>Su edad: <input type="text" name="edad" /></p>
 <p><input type="submit" /></p>
</form>
```

4.4 Entornos de desarrollo

Los framework son conjuntos de librerías, y entornos completos de desarrollo. Estos entornos, en el caso de PHP pueden emular algunas características de los contenedores de aplicaciones de Java, y facilitan la creación de portales de aplicaciones, gestores de contenidos, etc.

Los IDE son Entornos Integrados de Desarrollo, básicamente software donde escribes el código fuente y que te permite depurar errores, sugerir utilidades, etc. Los IDE más usados y que permiten un aprendizaje rápido del uso de PHP son:

- **Netbeans:** Es gratuito (la versión básica de Sun a partir del 2010) y tiene soporte para muchos idiomas. Cuenta con una gran comunidad de desarrolladores que trabajan en un entorno de desarrollo integrado. Además, se puede utilizar con otros frameworks como Zend, Doctrine, Smarty y Symfony2. Contiene plantillas de código, autocompletado inteligente, sugerencias, arreglos rápidos y refactorización.

- **PHPStorm: No** es el más aclamado porque muchas de sus utilidades no son gratuitas. Aun así, la versión gratis tiene bastante versatilidad. Puede interactuar con los frameworks: (Symfony, Zend, Yii, CakePHP y Laravel) y con los principales CMS como Drupal, Wordpress o el híbrido Magento para eCommerce. Pero también, en la parte de scripting para generar código html destaca la incorporación de las utilidades tales como CSS, HTML5, JavaScript o TypeScript. Expedia, Yahoo, Cisco, Salesforce y Wikipedia utilizan PHPStorm como su editor de código**.**
- **Eclipse:** Eclipse PDT (PHP Development Tools) es otra de las opciones open source con mayor recorrido y fiabilidad. Posee una enorme comunidad de desarrolladores que trabajan en todo tipo de plugins necesarios para poner a Eclipse a la altura de otros IDE de primera clase como PHPStorms, NetBeans y Zend Studio. La configuración inicial no es sencilla, por lo que habrá que dedicarle tiempo para configurarlo al gusto de cada usuario. Como aspecto negativo, destacar que algunos desarrolladores se quejan de que es lento. Sin embargo, esto ha dejado de ser un problema, ya que los ordenadores con los que se trabaja en estos días son suficientemente potentes como para no notar diferencia de rendimiento con otros IDE's. Algunas de las características clave incluyen el resaltado de sintaxis, ayuda de código, formateador de código, refactorización, plantillas de código, navegación de código, depuración de PHP, validación de sintaxis , además de la gran comunidad de usuarios**.**

Otros IDE de propósito general que se pueden adaptar para PHP son Atom, Aptana o Sublime Text.

En cuanto a entornos disponibles con características avanzadas cabe mencionar:

- **Symfony 5.1.** Se trata de un completo y potente entorno de desarrollo web basado en MVC, con una gran comunidad de desarrolladores y proyectos que lo soporta. Se basa en diversas herramientas que pueden utilizarse en conjunción o por separado, como Doctrine (ORM de acceso a datos), y Twig (gestor de plantillas web).
- **Laravel 7.23:** Es el que mayor proyección tiene en los últimos años, superando a Symfony en lo que se refiere a proyectos nuevos. La pieza de software para ORM se denomina Eloquent, y la gestión de las plantillas se realiza mediante Blade.

- **Zend Studio 13.6.** Es un framework comercial que junto con su entorno de desarrollo ayuda al desarrollo rápido de aplicaciones. Dispone de su propio servidor web.

Otros entornos: CodeIgniter, CakePHP, Yii, FuelPHP

5 CGI. Funcionamiento. Pros y contras

Common Gateway Interface (CGI) o interfaz de entrada común es una interfaz de los servidores web que permite intercambiar de manera estandarizada los datos entre los servidores y las aplicaciones externas.

Se encuentra entre las tecnologías de interfaz más antiguas de Internet, aunque su uso sigue muy extendido en la actualidad. Cuando se usa CGI no es necesario que todo el contenido de la página HTML esté disponible en el servidor, ya que se genera de forma dinámica cuando el usuario realiza la petición correspondiente a página web. Entonces, estos datos, en vez de enviarse directamente al servidor web, deben procesarse primero.

Este procesamiento no se lleva a cabo en el servidor, sino mediante un software externo (o un script CGI). Luego de procesarse y mediante la interfaz CGI, el programa transfiere los datos al servidor, que mostrará entonces el resultado de los cálculos en un documento HTML.

Por lo general, las aplicaciones CGI suelen almacenarse en su propio directorio en el servidor web.

El script CGI puede escribirse en gran variedad de lenguajes de programación. Common Gateway Interface garantiza que, al margen del lenguaje utilizado, el servidor web y el script puedan comunicarse entre sí.

5.1 Funcionamiento

A continuación se resume la forma en la que actúa CGI indicando las tareas que realiza el servidor y el modo de activación del CGI.

- El servidor web recibe una petición de un cliente sobre una URL que contiene la invocación al CGI.
- Seguidamente, el servidor prepara el entorno para ejecutar la aplicación externa a través del CGI. Esta información de entorno son básicamente variables que proceden mayoritariamente del cliente.
- El servidor ejecuta la aplicación y obtiene la salida.
- La aplicación realiza su función genera un objeto MIME (*) y la escribe en su salida estándar.
- Finalmente, cuando la aplicación finaliza, el servidor envía la información producida, junto con información propia, al navegador del cliente, que estaba a la espera.

Nota (*): Hay que indicar que un objeto MIME incluye objetos multimedia como audio o vídeo. Se deja a la aplicación el aviso de qué tipo de objeto MIME se ha generado mediante el campo CONTENT_TYPE.

Cómo se activa el CGI en cuanto a temas de petición o respuesta HTTP por el servidor:

- El cliente solicita la invocación de un CGI, ya sea de manera involuntaria (se envía únicamente información de cabecera) o de forma explícita (al completar un formulario web).
- En el **formulario web** hay parejas del tipo variable/ valor. El método de HTTP especificado en el formulario puede ser GET o POST (aunque se recomienda este último).
- En el archivo de configuración del servidor web se indica un directorio ***cgi-bin***, donde se pueden ejecutar los programas (aunque pueden haber otros ficheros o programas a los que se puede acceder tanto por parte del servidor como de sus aplicaciones CGI).
- El usuario cliente, cuando pulsa el botón de tipo SUBMIT en el formulario y dependiendo del método (GET/POST) **construye un mensaje que contiene la información del formulario en la cabecera en el caso de GET o en el cuerpo del mensaje para POST**.
- El mensaje se envía al servidor web, y, como ya hemos comentado antes, **añadiendo información propia del cliente que el propio navegador conoce** (como puede ser

información sobre cookies que veremos en el correspondiente capítulo más adelante). El cliente, tras enviar este mensaje, queda a la espera de recibir el objeto MIME respuesta del servidor web.

- El servidor web recibe el mensaje con la petición y lo trata (si hubiera un error por ejemplo), o bien activa el programa CGI: **El servidor compara la información que recibe del mensaje con la que sabe por su fichero de configuración**, determinando así la validez de la petición.
- El servidor web se cuestiona **aspectos tales como si existe esa URL o si el cliente tiene los permisos** para acceder a la información que solicita. Si, como decimos, no hay errores, el servidor web prepara la información que enviará a la aplicación CGI. Si el método HTTP requerido es GET, la información procedente del formulario (parejas variable=valor) se definen en la variable QUERY_STRING. Si se trata de POST, la información se coloca en la entrada estándar del CGI. El servidor posteriormente pone en funcionamiento el CGI y espera a que acabe.

Cómo ejecuta la aplicación el CGI:

- El script CGI examina las variables de entorno. Comprueba y se adapta según el método sea GET o POST en la variable **REQUEST_METHOD**: si se tratara de GET, la información estará en QUERY_STRING, mientras que si se trata de POST, se tomará la entrada estándar.
- Se construye un objeto MIME que se enviará al cliente indicando el tipo de objeto mediante la variable **CONTENT_TYPE**: tipo/subtipo.
- El servidor web añade a su respuesta del CGI una cabecera indicando su tamaño (**CONTENT_LENGTH**).
- El cliente recibe la respuesta interpretada y se puede visualizar en el navegador.

5.2 Paso de información al CGI

Como ya se comentó, los scripts de CGI pueden escribirse en cualquier lenguaje de programación que produzca un archivo ejecutable como por ejemplo C, C++, Perl, Java, Visual Basic. Incluso se escriben CGIs en lenguajes más antiguos como Cobol. La única restricción es que el intérprete del CGI pueda recibir los parámetros en forma de texto, por lo que el lenguaje utilizado debe poder manejar cadenas de caracteres. Para este menester es ideal Perl.

Existen tres métodos para transmitir los datos del servidor web al script CGI:

- QUERY_STRING: Se utiliza para la información de las peticiones de los usuarios.
- PATH_INFO: Este método envía información contextual sobre la URL en cuestión.
- Stdin: con este método, se procesan otras solicitudes del usuario, generalmente de tipo visual.

Además, como veremos en la tabla de la página siguiente, hay más variables que modulan tanto las peticiones como las respuestas.

El navegador del cliente envía la solicitud con información en la variable QUERY_STRING al servidor web. Este la analiza junto con el contexto (PATH_INFO) y determina si es válida o no y la envía al script CGI para su procesamiento. Al terminar el procesamiento, el servidor web formatea la respuesta y la envía al navegador web del cliente formateada según STDIN.

La siguiente tabla muestra las variables de entorno que pasan por los programas CGI:

Variables específicas de	Variable	Utilidad
El Servidor	SERVER_SOFTWARE	Nombre y versión del servidor web
El Servidor	SERVER_NAME	nombre de equipo del servidor, puede ser una dirección IP
El Servidor	GATEWAY_INTERFACE	versión CGI
Petición	SERVER_PROTOCOL	Versión HTTP
Petición	SERVER_PORT	Puerto TCP
Petición	REQUEST_METHOD	Nombre del método HTTP
Petición	PATH_INFO	Sufijo de la ruta
Petición	PATH_TRANSLATED	Ruta completa del servidor, si PATH_INFO está presente
Petición	SCRIPT_NAME	Ruta relativa al programa. Ejemplo: /cgi-bin/script.cgi.
Petición	QUERY_STRING	la parte del URL después del carácter ?. la cadena de consulta puede estar compuesta de *nombre=valor separados por el carácter & (ejemplo: var1=val1&var2=val2…) que se utiliza para enviar datos de un formulario web usando el método GET
Petición	REMOTE_HOST	Host del cliente
Petición	REMOTE_ADDR	IP del cliente
Petición	AUTH_TYPE	Tipo de autenticación del cliente
Petición	REMOTE_USER	Sirve para ayudar en la autenticación del cliente
Petición	REMOTE_IDENT	Para identificar algunas conexiones TCP
Petición	CONTENT_TYPE	Tipo de contenido de la aplicación
Petición	CONTENT_LENGTH	Tamaño de los datos de entrada

Tabla 1: Variables del CGI

Las variables que pasan por el agente del navegador del usuario del cliente (HTTP_ACCEPT, HTTP_ACCEPT_LANGUAGE, HTTP_USER_AGENT, HTTP_COOKIE etc.) contienen valores de sus correspondientes cabeceras HTTP por lo que sus datos no cambian al pasar por el CGI.

5.3 Usos de CGI en páginas web

A continuación referiremos algunos usos típicos de los scripts CGI con interés estadístico posterior.

- **Cesta de la compra:** Cuando un cliente añade algún producto a la cesta de la compra de una tienda online, el script CGI procesa esos datos y luego los envía al servidor.

- **Comentarios:** Cuando un usuario quiere dejar comentarios sobre el funcionamiento de un servicio o cualquier consulta, al hacer clic en el botón "Enviar" (en inglés Submit), el texto se transmite al script CGI y éste ,a su vez, lo reenvía al servidor.

- **Formularios:** Son muy versátiles, sirven tanto para enviar una propuesta en una web de trabajo como para reservar una plaza en un curso. Cuando se introducen los datos son procesados primero por un CGI antes de transmitirse al servidor.

- **Estadísticas de páginas web: Información de interés sobre las páginas web como el tráfico que tienen, la tecnología que lo respalda también recurre al CGI en muchos casos.**

- **Server Side Includes**: Mediante un CGI se permite cargar el contenido dinámicamente en la página web en formato de texto.

- **Pruebas de software:** A través del navegador, los desarrolladores pueden utilizar scripts CGI para probar la funcionalidad de las aplicaciones online externas para páginas web.

5.4 Ventajas y desventajas de su uso

A pesar de ser una técnica muy antigua, se sigue utilizando CGI en el desarrollo de páginas web a pesar de que presenta algunos inconvenientes.

A continuación sopesaremos las ventajas e inconvenientes:

Ventajas:

- Forma sencilla y eficaz de generar contenido dinámico.
- No se almacena en el servidor, de modo que no le incrementa estos recursos.
- Es compatible con casi cualquier lenguaje de programación.
- Es gratuito y un estándar amplio y disponible en cualquier momento.

Desventajas:

- Aunque la carga del servidor se reduce, **el tiempo de respuesta de las aplicaciones CGI se alarga mucho en algunos casos**, porque los programas deben volver a ejecutarse con cada nueva solicitud. Esto puede suponer mucha carga y que incluso se lleguen a rechazar peticiones.
- **Amenazas para la seguridad de la información**, ya que, a través del CGI los programas externos tienen acceso a todos los datos del servidor web. Por lo tanto, deben establecerse unas restricciones muy cuidadosas para evitar que los scripts CGI produzcan problemas.

Alternativas:

Se han desarrollado otras tecnologías de interfaz basadas en CGI, pero mitigando el principal inconveniente de que los scripts deban volver a ejecutarse con cada nueva solicitud del usuario.
A continuación las alternativas existentes:

- **ASP** (Active Server Pages): Aunque fue desarrollado originalmente por Microsoft para sus propios servidores, ahora está disponible para muchos otros. El intérprete de ASP está integrado en el servidor web, por lo que no es necesario invocar nuevos procesos cuando se utiliza. Los comandos ASP además se pueden escribir directamente en las páginas HTML. Además, también es compatible con varios lenguajes de programación.

- **PHP**: Es uno de los lenguajes de scripting más utilizados en Internet en la actualidad. Ofrece una funcionalidad muy similar a la de CGI pero el intérprete en este caso está integrado directamente en el servidor web.
- **ColdFusion**: Aunque se desarrolló originalmente para Windows, está disponible para varias plataformas Unix. El intérprete de ColdFusion está también integrado en el servidor web. Las páginas HTML se pueden modificar con etiquetas predefinidas o con elementos de control personalizados. Además, ColdFusion ofrece a los desarrolladores una serie de funciones estándar para incluir.
- **FastCGI:** Permite que las solicitudes dinámicas del servidor web pueden procesarse directamente a través de una interfaz escrita en el lenguaje Perl sin tener que iniciar un nuevo proceso para ello. Es compatible con CGI y con gran variedad de servidores web.

5.5 Invocación del CGI en HTML

Supongamos que queremos embeber en un HTML una referencia a un CGI desde otra página web.

Para ello usaríamos:

- Usando la etiquetas de HTML de la dirección del CGI, referencias, imágenes, etc. :

  Texto

  <img src="direccion_del_CGI">

- Usando un script de JavaScript o usando CSS. Por ejemplo: **<script src="/cgi-bin/script.cgi?NumberOfFiles&datadir" type="text/javascript" language="JavaScript"> </script>**

- Usando un formulario web (lo más habitual)

  <form action="direccion_del_CGI" > <!--

6 Cookies HTTP

En el mundo actual en el que vivimos, el consumo de contenidos de Internet por parte de los usuarios suele conllevar una contraprestación que muchos todavía ignoran: ***"Cuando el servicio es gratuito, el pago son tus datos".*** Los datos constituyen lo que se denomina el "oro azul" del siglo XXI y su explotación es todo un arte y una ciencia (Big Data).

Los datos de los usuarios y su comportamiento en la web se usan, entre otros, para lo que se denomina "consumerización" o perfilado/segmentación de usuarios. De esto modo, se pueden ofrecer ofertas comerciales lo más personalizadas posibles. De este modo, se intenta persuadir al usuario de que compre ciertos productos con los que luego se sufraga el servicio "totalmente gratuito".

Para poder obtener la información de este comportamiento de interacción de los usuarios en Internet con los sitios que se visitan, **se utiliza desde hace mucho tiempo un recurso que son las cookies**.

Una cookie HTTP, cookie web o cookie de navegador es uno o más pequeños ficheros de datos que un servidor web envía al navegador web del usuario y que este acepta para poder seguir interactuando con el sitio web. El navegador guarda estos datos y los envía al servidor web cuando éste los requiere.

El navegador web se puede configurar de modo que acepte o no y de qué modo las peticiones de cookies por parte de los servidores web. En la siguiente figura se muestra la configuración de privacidad y seguridad del navegador Chrome y cómo se puede configurar el manejo de cookies.

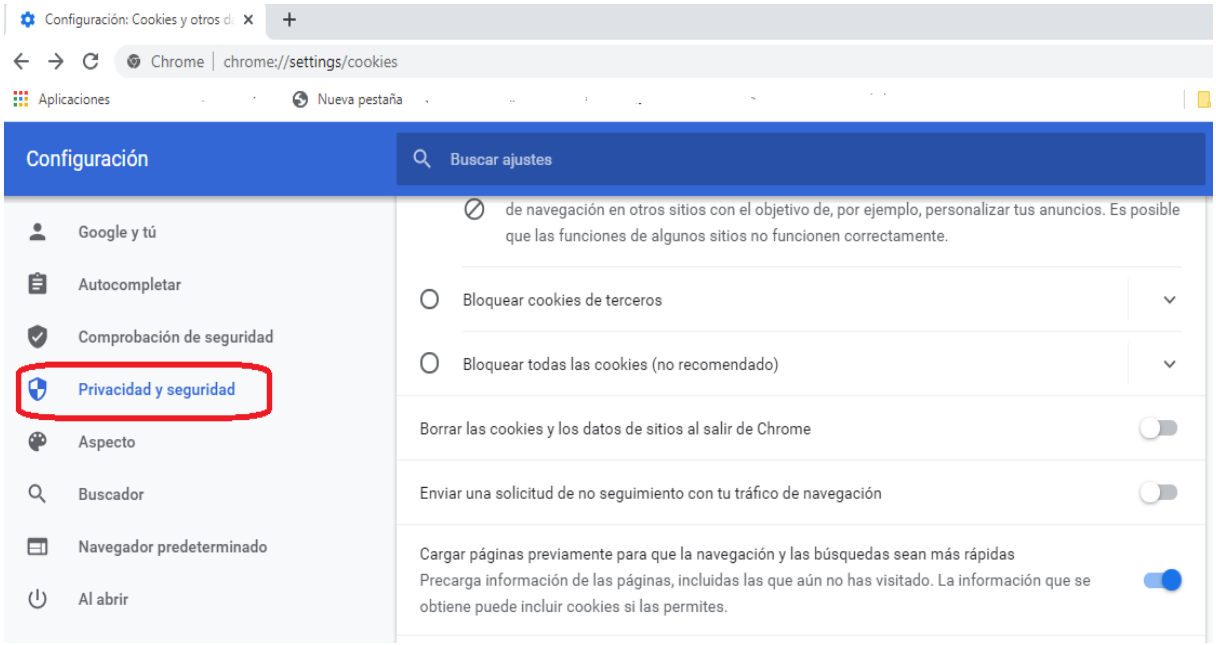


Figura 6: Configuración de las Cookies en Chrome. Fuente: Elaboración propia.

Entre otros usos las cookies permiten advertir al servidor web que ya se ha abierto sesión anteriormente con el servidor de modo que no es preciso iniciar nueva sesión (cookies de sesión). También permiten recordar la información del estado de dicha sesión ya que el protocolo HTTP es un protocolo sin estado.

6.1 Tipos de cookies

Dependiendo de quién sea el que gestiona el servidor o dominio desde donde se envían las cookies podemos distinguir entre:

- **Cookies propias:** son aquellas que se envían al equipo terminal del usuario desde un equipo o dominio gestionado por el propio editor y desde el que se presta el servicio solicitado por el usuario.
- **Cookies de terceros:** son aquellas que se envían al equipo terminal del usuario desde un equipo o dominio que no es gestionado por el editor, sino por otra entidad que trata los datos obtenidos través de las cookies.

Hay que hacer notar que no se pueden considerar cookies propias cuando se envían cookies desde un sitio web pero la finalidad no es un servicio propio sino de un tercero (por ejemplo, si en nuestra web del servicio hemos incluido un anuncio publicitario y es éste el que quiere información sobre los visitantes del sitio principal para luego captarlos como potenciales clientes.

Dependiendo de su duración:

- **Cookies Zombie**: Tienen la capacidad de regenerarse aunque hayan sido borradas del navegador.
- **Cookies Seguras**: Almacenan la información cifrada para impedir que los datos guardados puedan ser objeto de ataques contra la confidencialidad. Se usan con conexiones HTTP.
- **Cookies de Sesión**: Se generan y destruyen con la sesión de usuario.
- **Cookies Persistentes**: Son las más frecuentes, algunas duran más de un año incluso (por ley algunas hasta dos años), rastrean todos los procesos del usuario en la web en cuestión durante un período determinado. Se pueden borrar a través de la configuración del navegador.

Dependiendo de su uso o finalidad podemos distinguir entre:

- **Cookies técnicas:** son aquellas necesarias para seleccionar opciones o servicios en la web visitada y acceder a funciones o servicios de dicha web como, por ejemplo, controlar el tráfico y la comunicación de datos, identificar la sesión, acceder a partes de acceso restringido, recordar los elementos que integran un pedido (carrito o cesta), realizar el proceso de compra de un pedido, gestionar el pago, controlar el fraude vinculado a la seguridad del servicio, realizar la solicitud de inscripción o participación en un evento, contar visitas a efectos de la facturación de licencias del software con el que funciona el servicio (sitio web, plataforma o aplicación), utilizar elementos de seguridad durante la navegación, almacenar contenidos para la difusión de vídeos o sonido, habilitar contenidos dinámicos (por ejemplo, animación de carga de un texto o imagen) o compartir contenidos a través de redes sociales.
  También pertenecen a esta categoría, por su naturaleza técnica, aquellas cookies que permiten la gestión, de la forma más eficaz posible, de los espacios publicitarios embebidos en ella, siempre que no se recopile información de los usuarios con fines distintos, como puede ser personalizar ese contenido publicitario u otros contenidos para terceros ajenos.

- **Cookies de preferencias o de personalización**: Permiten una experiencia de usuario diferente en función de sus preferencias o de ciertos parámetros del navegador (idioma, número de resultados a mostrar por pantalla en una búsqueda, etc.). Si las opciones las ha seleccionado el propio usuario, como veremos luego, están exentas del consentimiento del usuario.

- **Cookies estadísticas (análisis o medición):** Son aquéllas que permiten cuantificar el número de usuarios, el impacto o "hits" de los anuncios mostrados y así realizar la medición y análisis estadístico de la utilización que el usuario hace del portal web. A destacar Google Analytics, servicio analítico de web prestado por Google y que se puede incorporar a nuestros servidores.

- **Cookies de publicidad comportamental:** Son las más "peligrosas" desde el punto de vista del riesgo para la privacidad del usuario ya que almacenan información de su comportamiento a través de la observación continuada de sus hábitos de navegación, lo que permite desarrollar un perfil específico sobre el mismo (consumerización) para mostrarle publicidad particularizada a lo que se cree son sus gustos o necesidades.

6.2 Cookies y estadísticas de los sitios web

Como acabamos de mencionar, las cookies estadísticas (análisis y medición) permiten recabar información para analizar de forma estadística el comportamiento de uso del sitio web por parte de los usuarios. A continuación introduciremos brevemente el uso de Google Analytics

Los pasos grosso modo para instalar Google Analytics en un sitio web son:

- Crear una cuenta en Google Analytics (si no se dispone ya de una corporativa).
- Crear una nueva propiedad GA4 y universal Analytics en el menú de Google Analytics (Administrar=> Crear Propiedad).
- Crear un flujo de datos para la medición de la web. (Una propiedad puede tener varios flujos canales de datos procedentes de distintas plataformas, hay que seleccionar de dónde tomaremos la información.)

- Copiar el ID de medición. Desde el menú de la propiedad GA4 creada.
- Crear una etiqueta de configuración de Google Analytics en la página web a medir.
- Pegar el ID de medición en la etiqueta para que se active en todas las páginas.
- Hacer las pruebas con la nueva etiqueta y corregir lo que no guste.
- Publicar finalmente en la web en producción los cambios realizados en Google Tag Manager.

Algunas mediciones que se suelen hacer:

- **Informes de Audiencia:** Permite conocer datos agregados de las personas que visitan el sitio web. Se descompone a su vez en los siguientes informes:

    - Visión general: Características básicas de los visitantes tales como ubicación geográfica.
    - Intereses: Información sobre gustos y aficiones generales de los visitantes (pero Google Analytics solo puede detectar esta información de un cierto porcentaje de usuarios, no de todos).
    - Comportamiento: indica, por ejemplo, la frecuencia de las visitas, el tiempo de interacción de los usuarios y hasta información sobre si se accede con algún dispositivo móvil.
    - Tecnología: Información sobre el navegador web, los sistemas operativos (a través de la huella que ofrece el navegador web "fingerprinting" e incluso los proveedores de acceso Internet que utilizan los usuarios.

- **Informes de Adquisición:** Permite saber cómo los usuarios llegan al sitio web. Para esto, es importante conocer los diversos canales que alimentan y proporcionan datos a Google. En términos generales son los siguientes:

    - Canal orgánico: Motores de búsqueda: Google, Yahoo, Bing, etc.
    - Canal directo: Usuarios que teclean directamente la URL en la barra de direcciones de su navegador.
    - Canal referencial: Son aquellos links entrantes que hacen referencia al sitio web. Es decir, engloba a los usuarios que accedieron desde otro sitio web.
    - Canal social: Muestra todo el tráfico que llega desde redes sociales: Facebook, Twitter, Instagram, etc.

- **Informes de Comportamiento:** Permite saber cómo los usuarios interactúan con el sitio web. En particular informa sobre qué páginas (del sitio web) visitan y qué ruta han seguido para llegar a ellas. Contienen dos subinformes interesantes:
    - Flujo del comportamiento: Indica la ruta que siguen los usuarios desde una página o un evento al siguiente. Este informe puede ayudar a descubrir qué contenido mantiene la interacción de los usuarios con el sitio web y qué otro, por el contrario, está causando que los usuarios abandonen el sitio.
    - Analítica de página: Evaluación visual del modo en que los usuarios interactúan con el sitio web. Sirve para saber el diseño es adecuado para la interacción necesaria, si los usuarios advierten el contenido que necesito que vean, si encuentran visualmente lo que necesitan o si los botones o los links están visualmente bien resaltados

- **Informes de Conversiones**: Las conversiones son las acciones que llevan a cabo los visitantes de un sitio web que forman parte de los objetivos de los propietarios del sitio (por ejemplo, si se visita una tienda online, se desea que el visitante finalice su periplo por la web adquiriendo productos). Para medir esta variable se usa una fórmula denominada "tasa de conversión" que obtiene el porcentaje de los objetivos conseguidos entre el total de visitas del sitio. Para poder utilizar este tipo de informes es preciso contar con una versión avanzada de Google Analytics. En su menú Conversiones podremos acceder a tres opciones: Objetivos, Comercio electrónico y Embudos multicanal.

En cuanto a la arquitectura de Google Analytics, es preciso mencionar que usa tres bibliotecas de JavaScript para medir el uso de los sitios web: gtag.js, analytics.js y ga.js.

Las cookies que se crean con Google Analytics se muestran en la siguiente tabla.

Nombre de la Cookie	Duración Predeterminada	Uso estadístico
_ga	2 años	Se usa para distinguir a los usuarios.
_gid	24 horas	Se usa para distinguir a los usuarios.
_ga_<container-id>	2 años	Se usa para mantener el estado de la sesión.
_gac_gb_<container-id>	90 días	Incluye información relacionada con la campaña. Si has vinculado tus cuentas de Google Analytics y Google Ads, las etiquetas de conversión en sitio web de Google Ads leerán esta cookie, a menos que la inhabilites.
_gat	1 minuto	Se usa para limitar el porcentaje de solicitudes. Si se ha implementado Google Analytics mediante *"Google Tag Manager",* esta cookie se llamará _dc_gtm_<property- id>.

Nombre de la Cookie	Duración Predeterminada	Uso estadístico
AMP_TOKEN	De 30 segundos a 1 año	Incluye un tóken que se puede utilizar para recuperar un ID de cliente del servicio de IDs de cliente de AMP. Otros posibles valores indican inhabilitaciones, solicitudes en curso o errores obtenidos al recuperar un ID del servicio de IDs de cliente de AMP.
gac<property-id>	90 días	Incluye información de la campaña relativa al usuario. Si has vinculado tus cuentas de Google Analytics y Google Ads, las etiquetas de conversión en sitio web de Google Ads leerán esta cookie, a menos que la inhabilites.

Tabla 2: Cookies de Google Analytics

Las cookies generadas se imputarán al dominio más particularizado. Por ejemplo, si el sitio web es ***"blogdepepe.midominio.co.es"***las cookies se asignan al dominio ***"midominio.co.es"***. De este modo, se puede hacer el seguimiento de los usuarios en varios subdominios sin tener que llevar a cabo ninguna configuración adicional

6.3 Seguridad, privacidad, LSSI y datos protegidos LOPD

La Ley de Servicios de Sociedad de la Información (LSSI), regula los derechos de los usuarios de sitios web y las obligaciones de los proveedores de servicios y contenidos. Entre los derechos de los usuarios se encuentra la privacidad y el poder tener el control de cómo se manejan sus datos por parte de los propietarios de los sitios web como por parte de terceros involucrados.

En lo que afecta a la privacidad y el poder de decisión del usuario en lo que a las cookies se refiere:

El artículo 22.2 De la LSSI dice que: ***Los prestadores de servicios podrán utilizar dispositivos de almacenamiento y recuperación de datos en equipos terminales de los destinatarios, a condición de que los mismos hayan dado su consentimiento después de que se les haya facilitado información clara y completa sobre su utilización, en particular, sobre los fines del tratamiento de los datos, con arreglo a lo dispuesto en la LOPD.***

La Directiva europea: ***2009/136/CE de 25 de noviembre de 2009***, estableció en su día que se debería poder escoger libremente por parte del usuario la aceptación o no de las cookies que no fueran de tipo técnico y estrictamente necesarias.

Entre las cookies no necesarias se indicó que eran las siguientes:

- Cookies de seguimiento.
- Cookies de segmentación.
- Cookies de análisis.
- Cookies de redes sociales.

Esta directiva europea se traspuso en España en la conocida "Ley de Cookies" (Real Decreto-ley 13/2012 de 30 de marzo). Un año más tarde la Agencia Española de Protección de Datos (AEPD) elaboró una "***Guía sobre el uso de cookies"*** (consultar bibliografía básica recomendada).

En dicha guía resumida, se viene a decir que las cookies técnicas están exentas del artículo 22.2 de la LSSI cuando sean imprescindibles para la funcionalidad del servicio prestado. Sin embargo, **si estas cookies se utilizan también para finalidades no exentas (por ejemplo, para fines publicitarios comportamentales), quedarán sujetas a dichas obligaciones.**

Además, en esta guía se remarcan tres aspectos fundamentales:

- **Obligación del consentimiento** informado (antes de comenzar a navegar por el sitio web y aceptar las cookies).
- **Transparencia a la hora de informar sobre el uso** de las cookies (para qué se usan y poder decidir de forma particularizada sobre qué usos se aceptan o no, en línea con el actual RGPD).
- **Revalidación periódica del consentimiento** (como máximo cada dos años).

Uno de los riesgos más graves del uso de cookies por parte de los sitios web es el "robo o secuestro de las cookies de sesión" por parte de terceros. Esto se conoce con el término en inglés "Cookie hijacking". De este modo, un tercero malicioso nos puede suplantar con lo que esto comporta de gravedad. Entre otros consejos, los desarrolladores del sitio web deben usar siempre cifrado SSL/TLS al menos cuando requieran las credenciales del usuario y, por supuesto, la asignación de la cookie de sesión debe ser mediante algoritmos y métodos que impidan que la cookie se quede fijada (persistente) sino que se regenere con cada nueva sesión que inicie el usuario. Tampoco deberá ser predecible.

Otro problema de seguridad de los sitios web también grave es el Cross-Site-Scripting (XSS): Si un sitio web contiene esta vulnerabilidad, un atacante puede realizar diversos tipos de ataques tales como redirigir a otro link ajeno al sitio web y con intenciones de phising o a un link para que se descargue malware, y todo basado en que el usuario piensa que el sitio web es confiable y seguro. En este segundo caso y para prevenir en lo posible estos ataques, las cookies del lado del servidor se deben enviar con el flag **HTTP_ONLY en la cabecera de la petición del lado del servidor.**

Por ejemplo, en JavaScript sería algo así como:

Set-Cookie: sessionid=QmFieWxvbiA1; HttpOnly

Figura 7: Secuestro de sesión. Fuente: https://losindestructibles.wordpress.com

Por último destacar que **existen algunos navegadores web que permiten una total privacidad para evitar que la información del usuario llegue en exceso o sin ser solicitada al servidor web**. Como ejemplo mencionaremos Epic Browser, con el mismo motor que Chrome (chromium).

La navegación de incógnito que permiten algunos navegadores web como Chrome permite controlar qué datos se exponen, pero no garantiza el anonimato. Es decir: lo que evita es que nuestro

navegador almacene datos que identifican nuestro comportamiento en la web y que se almacenan en la caché, cookies, historial de navegación etc.

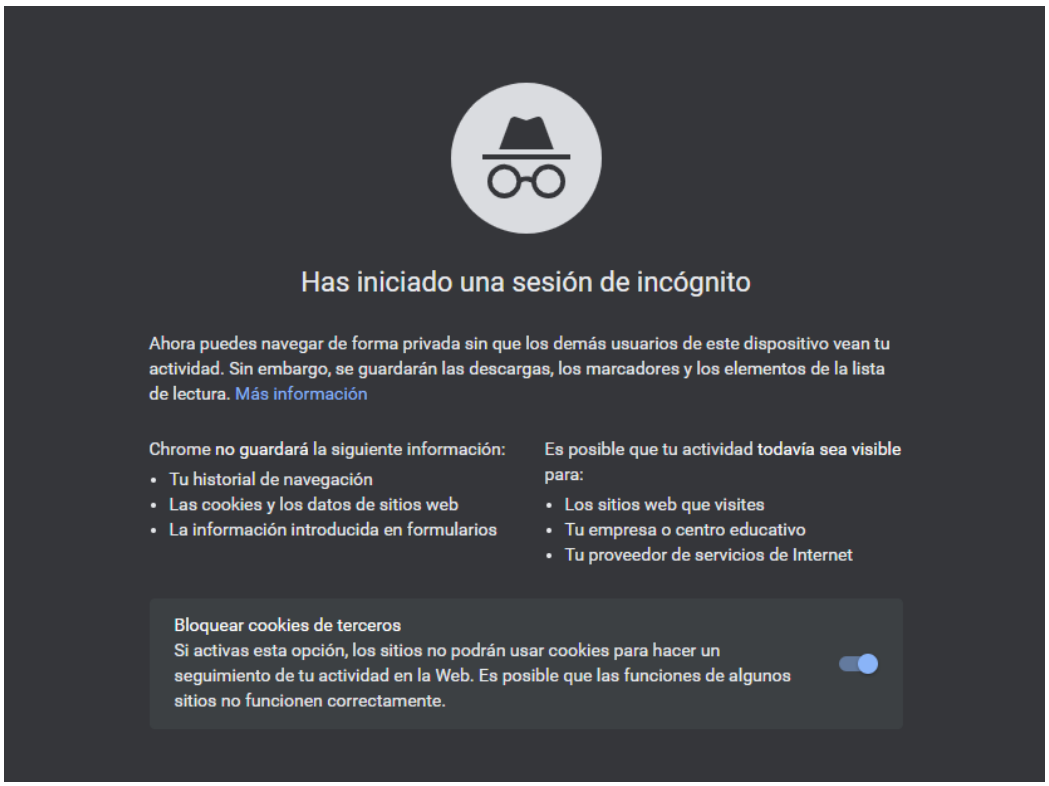


Figura 8: Navegación de incógnito en Chrome

Una observación final: **No hay que confundir privacidad con anonimato**. La privacidad es el control sobre qué y a quién mostramos nuestros datos pero el anonimato pretende no mostrar ningún dato en absoluto en la navegación web (algunos relacionan esto con deseos de impunidad cuando se realizan ciertas acciones en la web) y sólo se consigue usando navegadores específicos para ello tales como los usados para entrar en la Deep Web (TOR Onion o similares).

7 RESUMEN ESQUEMÁTICO

- Los lenguajes de scripts permiten **dotar de dinamismo** a las webs. Pueden incluirse en el servidor web o para ser ejecutados en el navegador de los clientes.
- Los lenguajes de scripting representan en la actualidad aproximadamente un tercio de todos los lenguajes de programación más utilizados en el mundo.
- Los lenguajes de scripts, a diferencia de la mayoría de los lenguajes de programación **son interpretado**s, de modo que no necesitan de ser compilados previamente para su ejecución.
- Un script de cliente es un programa incrustado en el código HTML del servidor web y que es interpretado y ejecutado por el navegador del usuario.
- **Los scripts del lado del servidor sirven para realizar tareas más complejas** y cercanas a la lógica de negocio tales como transacciones y su registro en base de datos.
- A diferencia de los scripts del lado cliente, en los scripts del lado del servidor, el código fuente de los scripts permanece oculto para cliente.
- Para poder ejecutar los scripts del lado servidor es preciso que el servidor y el cliente tengan establecido un canal de comunicación síncrona. En el caso de los scripts del lado cliente, esto no es preciso, se pueden ejecutar de forma asíncrona.
- **En las primeras décadas de Internet, los scripts del lado del servidor necesitaban del uso de una interfaz CGI para poder comunicarse con los clientes**. Eso era así porque se basaban en lenguajes imperativos como C. En la actualidad con el uso de lenguajes de scripting como PHP o Perl, CGI ya no es necesario.
- **ECMAScript es el estándar de normalización de lenguajes de scripts del lado cliente**. La primera versión data de 1996 y fue la ECMA-262. La última versión es la numero 11 que data del año 2020.
- ECMAScript inicialmente fue una generalización del JavaSript inicial de Netscape, aunque luego ha ido evolucionando. No es orientado a objetos aunque se pueden usar objetos. Contiene una definición de sintaxis, operadores y tipos de objetos que se pueden utilizar.
- **El código JavaSript como se embebe en el HTML correspondiente usando la etiqueta <script> indicando que el lenguaje es JavaScript.**
- **JavaSript** no es orientado a objetos, pero usa objetos. Los tipos de datos que se usan son muy simples: Numéricos, Booleanos y de tipo cadena de literales, String. Además, no es necesario declarar los tipos de las variables.
- **La sintaxis de JavaScript se basa en funciones** (entre llaves y con argumentos en forma de tupla con paréntesis) **y variables** (no se definen sino que se les asigna directamente el valor).
- **JScript** fue desarrollado por Microsoft, se incorporó en Internet 3.0 y aunque comparte el estándar de ECMAScript, incorpora librerías y funciones típicas de ActiveX y similares de los años 90.
- **Actualmente se desaconseja el uso de JScript por motivos de seguridad.** Se puede deshabilitar en Windows 10. Se reemplazó por JScript.NET (que no es lenguaje de scripting).
- **VBScript** es similar a VBasic pero más simple. Está en desuso. Sólo funcionaba con Internet Explorer.
- **TypeScript** de Microsoft es software libre pero que se usa para ampliar JavaScript (algo así como framework de desarrollo) y adaptarlo a proyectos grandes. Incluye más funcionalidades como un compilador a código JavaScript y además amplia los tipos básicos de éste.
- **PHP es el principal lenguaje de scripting en el servidor que no precisa de uso de CGI**. Es de código libre.
- **Usando PHP se puede generar páginas de contenido dinámico (incrustando incluso JavaScript, CSS, etc.), manipular archivos de todo tipo, acceder a ficheros de base de datos, manejar sesiones de usuario y cookies, cifrar los datos, crear formularios, etc.**

- PHP se puede instalar tanto en Linux como en Windows, y el entorno de instalación precisa de la instalación previa de una base de datos.
- Los entornos de desarrollo más usados con PHP son Symfony, Laravel o Zend Studio.
- **PHP es muy usado para implementar gestores de contenidos** como Drupal o Magento.
- **CGI es sencillo de usar, compatible con casi cualquier lenguaje, gratuito y además no se almacena en el servidor.**
- El navegador del cliente envía la solicitud con información en la variable QUERY_STRING al servidor web. Este la analiza junto con el contexto y otras variables (como por ejemplo PATH_INFO) y determina si es válida o no y la envía (si es válida) al script CGI para su procesamiento. Al terminar el procesamiento, el servidor web recibe el resultado del CGI y formatea la respuesta para enviarla al navegador web del cliente, para lo cual utilizará el método STDIN.
- Las dos principales desventajas del uso de CGI son la penalización en el tiempo de respuesta y las amenazas para la seguridad del uso de este tipo de interfaces relativamente fáciles de interceptar y analizar.
- Una cookie HTTP, cookie web o cookie de navegador **es uno o más pequeños ficheros de datos que un servidor web envía al navegador web del usuario y que este acepta para poder seguir interactuando con el sitio web**. El navegador guarda estos datos y los envía al servidor web cuando éste los requiere.
- Según quién las requiere, las cookies pueden ser **propias** del servidor web o alojadas por **terceros.**
- **Según su uso las cookies pueden ser: Técnicas, de personalización, de estadísticas o de publicidad.**
- **Según su duración pueden ser:** Cookies Zombie (se regeneran solas)**,** Cookies Seguras (van cifradas durante toda la interacción con el sitio web), Cookies de Sesión (sólo duran la sesión del usuario) y Cookies persistentes (hasta dos años si no se borran).
- **El artículo 22 de la LSSI establece la obligatoriedad del consentimiento en aquellas cookies no técnicas. Además debe informarse de forma transparente y revalidar periódicamente el consentimiento.**
- **La Agencia Española de Protección de Datos** creó una Guía sobre el uso de Cookies conforme a la legislación de privacidad aplicable.
- **Uno de los peligros más importantes desde el punto de vista de los ataques de ciberseguridad es el llamado "secuestro de cookies**", que ocurre cuando el atacante intercepta la petición cliente-servidor en curso debido a que puede conocer los datos de la sesión en curso, por las vulnerabilidad que presenta. Para prevenir en lo posible estos ataques, las cookies además de diseñar de forma correcta la asignación de tókens de sesión de modo que no sean predecibles y de establecer conexiones cifradas, las peticiones del lado del servidor se deben enviar con el flag "HTTP_ONLY" en la cabecera de la petición.
- **Navegar en modo incógnito y el uso de ciertos navegadores** ayuda a **preservar la privacidad** pero no el anonimato. Éste sólo se garantiza mediante el uso de herramientas típicas de la Deep Web como el navegador del proyecto TOR. Al preservar la privacidad, se eliminan las cookies de rastreo y el historial de navegación, entre otros.

8 GLOSARIO

- **Applets (de Java):** Código que se incrusta en el código HTML de la página web. Cuando un Navegador carga una página Web que contiene un Applet, éste se descarga en el navegador Web del usuario y comienza a ejecutarse de forma cuasi independiente de la web (normalmente en otra ventana o recuadro y de forma interactiva con el usuario. Uno de los requisitos para el applet funcione es tener instalado en el navegador web la máquina virtual de Java (JVM).
- **Cookie**: Fichero o datos que el servidor envía al navegador del cliente y que éste incorpora, que sirve para rastrear el comportamiento o para adaptar la experiencia de usuario y opciones del cliente.
- **Cookie Hijacking**: Ver Hijacking.
- **Consumerización**: Acción de adaptar las ofertas y opciones de un servicio web a las preferencias del cliente.
- **CGI:** Common Gateway Interface. Interfaz para el paso de variables e información entre los programas del servidor y el navegador web del cliente.
- **CSS:** Cascading Style Sheets, "Hojas de estilo en cascada", que es un lenguaje de marcas (ver tema 7), enfocado a definir, crear y mejorar la presentación de un documento basado en HTML, al proveer de estilos y toda suerte de herramientas en el diseño de una web.
- **Google Analytics:** Paquete de software de análisis (de Google) de comportamiento de usuarios en la web que se puede instalar sobre un servidor web para introducir elementos de tracking y seguimiento del comportamiento de los usuarios mediante cookies específicas y análisis de las mismas. Se usa para obtener estadísticas tales como número de usuarios que han accedido, ruta que suelen seguir, acciones que suelen ejecutar preferentemente, etc.
- **Google Tag Manager:** Administrador de etiquetas para insertar código de referencia de Google Analytics y en general, de las cookies de Google.
- **G4:** Google Analytics 4, módulo generador de propiedades.
- **Hijacking:** Secuestro. Referido a cookies, Cookie Hijacking es un ataque a la seguridad de una web consistente en que el atacante puede conocer (por vulnerabilidades del diseño de la web) el identificador de sesión de un usuario y reproducir en paralelo dicha sesión en su navegador web, de modo que puede saber todo lo que está haciendo el usuario y robarle la información.
- **HTML:** HyperText Markup Language, hace referencia al lenguaje de marcado para la elaboración de páginas web.
- **JavaScript:** Lenguaje de scripts de lado cliente creado por Netscape. A día de hoy es el más extendido. Aunque utiliza objetos, no es orientado a objetos como sí lo es Java.
- **JScript:** Lenguaje de scripts de lado cliente creado por Microsoft que fue la competencia durante un tiempo de JavaScript. Ha caído en desuso y ha sido además supercedido por tecnología .NET (no scripting).
- **LOPD-GDD:** Ley Orgánica de Protección de Datos y Garantías de Derechos Digitales, Ley orgánica 3/2018 que interpreta el RGPD y lo incorpora a la normativa española.
- **LSSI: Ley 34/2002, Ley de Servicios de la Sociedad de la Información y Comercio Electrónico.** Regula entre otros aspectos, el consentimiento del usuario para que se utilicen sus datos con fines comerciales.
- **RGPD:** Reglamento General de Protección de Datos. Reglamento Europeo que regula las cuestiones relativas a la protección de datos de carácter personal y es de obligado cumplimiento para todos los Estados Miembros.
- **Script:** Programa muy simple que consiste en una secuencia de comandos con algunas variables. Los más populares son los scripts para sistemas operativos (batch, bash, powershell) o de lenguajes para tareas multipropósito (Python, Perl). En el tema hemos visto

los scripts para conferir dinamismo a las páginas web tales como JavaScript, JScript, PHP o los CGI.

- **Secuestro de Cookies (cookie hijacking):** Ciberataque consistente en la interceptación con fines maliciosos de la sesión en curso entre un cliente y un servidor web. Este problema es fruto de un mal diseño de la asignación de tókens de sesión. Se mitiga mediante el uso de la directiva HTTP_ONLY.
- **TypeScript:** Lenguaje de scripting compilado que usa de base JavaScript pero incorpora mejoras para grandes proyectos.
- **VBScript:** Lenguaje de scripting que toma como base Visual Basic. Sólo funciona con navegadores Internet Explorer y ha caído en desuso salvo en servidores web que usan ASP.
- **Widgets:** Gadgets (utilidades con representación gráfica) para mostrar en la web en cuestión utilidades disponible en el sistema tales como relojes y cronómetros, notas de texto, calculadoras, calendarios o información del tiempo.
- **XSS:** Cross-Site-Scripting, ataque de seguridad de la web consistente en

9 BIBLIOGRAFÍA BÁSICA

9.1 BIBLIOGRAFÍA BÁSICA

1. ASTIC. Temario para la preparación de oposiciones TIC A1. https://www.astic.es/opositor/temario
2. Preparatic. Material de apoyo para la preparación de oposiciones: http://www.preparatic.org/
3. Guía digital de IONOS sobre desarrollo web: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-son-los-lenguajes-de-scripting/
4. Página web de ECMA internacional para consultar el estándar: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/
5. Página oficial de PHP: https://php.net
6. Ejemplos sencillos con PHP: https://www.php.net/manual/es/indexes.examples.php
7. Guía de PHP de la universidad complutense: https://www.fdi.ucm.es/profesor/jpavon/web/33-PHP.pdf
8. Universidad Carlos III: Curso sobre lenguajes de scripting, uso de CGI: http://www.it.uc3m.es/mcfp/docencia/si/material/3_cgi_mcfp.pdf
9. Guía sobre las Cookies elaborada por la Agencia Española de Protección de Datos: https://www.aepd.es/sites/default/files/2020-07/guia-cookies.pdf

9.2 BIBLIOGRAFÍA PARA AMPLIAR EL TEMA

10. Diferencias entre VBScript y JavaScript: https://www.ijcait.com/IJCAIT/13/134.pdf
11. Web de Monografías: https://www.monografias.com/trabajos55/tutorial-de-jscript/tutorial-de-jscript.shtml
12. Ejemplos sobre JavaScript propuestos por la Universidad de Valencia : https://www.uv.es/jac/guia/jscript/javascr.htm
13. Guía para el Uso de las Cookies de Google Analytics en los sitios web, https://developers.google.com/analytics/devguides/collection/analyticsjs/cookie-usage?hl=es-419
14. Guía sobre Google Analytics para su instalación y primeros pasos: https://romualdfons.com/tutorial-guia-google-analytics/