Desarrollo Web en entorno Cliente: JAVASCRIPT

UNIDAD 1. SELECCIÓN DE ARQUITECTURAS Y HERRAMIENTAS DE PROGRAMACIÓN

1.1 FUNDAMENTOS DE JAVASCRIPT

1.2 EVOLUCIÓN Y CARACTERÍSTICAS DE LOS NAVEGADORES WEB

1.3 ARQUITECTURA DE EJECUCIÓN

1.4 LENGUAJES Y TECNOLOGÍAS DE PROGRAMACIÓN EN ENTORNO CLIENTE

1.5 INTEGRACIÓN DEL CÓDIGO CON LAS ETIQUETAS HTML

- 1.5.1 JavaScript en el mismo documento HTML
- 1.5.2 JavaScript en un archivo externo
- 1.5.3 JavaScript en elementos HTML

UNIDAD 1. SELECCIÓN DE ARQUITECTURAS Y HERRAMIENTAS DE PROGRAMACIÓN

1.1 FUNDAMENTOS DE JAVASCRIPT

El lenguaje que vamos a estudiar se llama JavaScript, pero puede que hayamos oído otros nombres que nos resulten similares como JScript (que es el nombre que le dio Microsoft).

https://www.thoughtco.com/javascript-and-jscript-whats-the-difference-2037681

DIFERENCIAS

La mayor diferencia ahora entre JavaScript y JScript son todos los comandos adicionales que soporta JScript que permiten el acceso a ActiveX y al equipo local. Estos comandos están diseñados para su uso en sitios de intranet donde conoce la configuración de todos los equipos y que todos ejecutan EDGE.

Todavía quedan algunas áreas donde JavaScript y JScript difieren en los medios que proporcionan para realizar una tarea en particular. Excepto en estas situaciones, *los dos lenguajes pueden considerarse equivalentes entre sí y, a menos que se especifique lo contrario, todas las referencias a JavaScript que usted vea también incluirán JScript.*

JScript es un invento de Microsoft para apartarse del lenguaje estándar y conseguir que las páginas sólo se puedan ver usando Windows, y de esta forma eliminar la libre competencia en materia de browsers.

El lenguaje Javascript es un estándar de ECMA y funciona en todos los browsers modernos.

Microsoft le dio el nombre de JScript para evitar problemas relacionados con la marca, pero no pudo evitar otros problemas surgidos por las incompatibilidades que su versión de JavaScript tenía con múltiples navegadores. Para evitar esas incompatibilidades, el W3C, diseñó el **DOM** (Modelo de Objetos del Documento), que se incorporaron a las versiones de Internet Explorer 6, Netscape Navigator, Opera 7 y Mozilla Firefox desde su primera versión.

A partir de 1997 este lenguaje se rige por un estándar denominado **ECMA**, que se encarga de gestionar las especificaciones de este lenguaje de script (da igual el nombre que reciba). En el documento ***ECMA-262*** es dónde se detallan dichas especificaciones:

http://www.ecma-international.org/publications/standards/Ecma-262.htm

Tanto JavaScript como JScript son compatibles con el estándar ECMA-262.

JavaScript se diseñó con una sintaxis similar al lenguaje C y aunque adopta nombres y convenciones del lenguaje Java, éste último no tiene relación con JavaScript ya que tienen semánticas y propósitos diferentes.

JavaScript fue desarrollado originariamente por Brendan Eich, de Netscape, con el nombre de Mocha, el cual se renombró posteriormente a LiveScript y quedó finalmente como JavaScript.

Hoy en día JavaScript es una marca registrada de Oracle Corporation, y es usado con licencia por los productos creados por Netscape Communications y entidades actuales, como la fundación Mozilla.

Aclarando las distintas versiones de **ECMAScript HISTORIA-EVOLUCIÓN**:
https://en.wikipedia.org/wiki/ECMAScript_version_history

Especificación ECMAScript → implementación diferente dependiendo del navegador.

ES6 → ES2015

ES7 → ES2016

ES8 → ES2017

Ecmascript2019 ES10 : Novedades https://ricardogeek.com/que-hay-de-nuevo-en-ecmascript-2019/ Las operaciones sobre estructuras de datos, son las que hacen esta versión realmente interesante de revisar y hacen que javascript sea un lenguaje bonito. Úsenlas con responsabilidad!

Otro enlace sobre novedades de Ecmascript2019 **https://javascriptfacil.com/ecmascript-2019/**

Ecmascript2020 ES11: https://pablomagaz.com/blog/un-vistazo-a-ecmascript2020

https://ricardogeek.com/que-hay-de-nuevo-en-ecmascript-2020/

https://medium.com/get-on-board-dev/2-nuevos-features-de-javascript-que-deber%C3%ADas-conocer-si-lo-usas-todos-los-d%C3%ADas-ecmascript-2020-83f48bb797aa

Ecmascript2021 ES12: https://cosasdigitales.com/articulos-diseno-web/es2021-es12-novedades-de-la-ultima-version-de-javascript/

Ecmascript2022 ES13: https://victorroblesweb.es/2022/04/26/javascript-es2022-novedades-y-mejoras-del-lenguaje-de-frontend-mas-usado/

https://matiashernandez.dev/blog/post/que-hay-de-nuevo-en-javascript-2022

Actualmente EcmaScript2023 ES14 (14 Edición de EcmaScript finalizado en junio del 2023) :

https://malcoded.medium.com/ecmascript-2023-70bbb93d4f9a

https://appmaster.io/es/news/javascripts-ecmascript-2023-aprobacion

https://appmaster.io/es/news/ecmascript-2023-nuevas-funciones-javascript-que-mejoran-las-matrices-y-las-claves-weakmap

https://tc39.es/ecma262/2023/

Aplicaciones Cross-Browser (multi-cliente)

Cuando hablamos de aplicaciones ***cross-browser***, nos estamos refiriendo a aplicaciones que se vean exactamente igual en cualquier navegador.

Como bien sabemos los navegadores son desarrollados por diferentes empresas de software, cada una con sus propios intereses y, desde siempre, han sido patentes las diferencias entre unos y otros. El ***W3C*** define estándares para HTML, CSS y JavaScript, pero muchas veces estas empresas interpretan el estándar de forma distinta, o incluso, a veces, agregan funcionalidades o etiquetas que no están contempladas ni permitidas en el estándar.

El W3C ha ido mejorando y actualizando los estándares, definiendo nuevos niveles del DOM, y por el otro lado, las empresas desarrolladoras de software también se van adaptando, cada vez más, a los estándares propuestos por el W3C.

La historia de cross-browser comenzó con la "guerra de navegadores" al final de 1990 entre Netscape Navigator y Microsoft Internet Explorer y, por lo tanto, también entre JavaScript y JScript (los primeros lenguajes de scripting implementados en estos navegadores respectivamente). Netscape Navigator era el navegador web más usado en ese momento, y Microsoft había sacado Mosaic para crear Internet Explorer 1.0. Nuevas versiones de estos navegadores fueron surgiendo rápidamente, y debido a la feroz competencia entre ellos, muchas veces se añadieron características nuevas, sin ningún tipo de coordinación o control entre fabricantes. La introducción de estas nuevas características a menudo tuvo prioridad sobre la corrección de errores, dando como resultado navegadores inestables, bloqueos, navegadores que no cumplen el estándar y fallos de ejecución, llegando incluso a provocar cierres accidentales de las aplicaciones o del navegador.

Durante todo ese tiempo los programadores de páginas web han sido los encargados de ir parcheando estas diferencias para conseguir que sus aplicaciones se ejecuten de la misma forma en unos u otros navegadores, independientemente de la versión o fabricante utilizado. ***Estas soluciones que se adaptan a cualquier tipo de navegador son las que se conocen como "soluciones cross-browser"***.

Las soluciones cross-browser no sólo se aplican a JavaScript, sino que también se pueden aplicar a otras tecnologías como CSS o, incluso, HTML. Lo que se busca por lo tanto es que, esas incompatibilidades o diferencias entre navegadores no sean apreciables por el cliente, y que la página web o aplicación funcione indistintamente en cualquier navegador sin producir fallos o efectos indeseados.

En Internet puedes encontrar múltiples páginas con tablas donde ver las incompatibilidades entre navegadores a nivel de, CSS 2, CSS 3, base del DOM, DOM HTML, eventos del DOM, etc. En estas tablas se muestran todas las características de cada tecnología, se ven las diferentes versiones de navegadores, y se indica si soportan o no, cada una de las características, propiedades, métodos, etc. A continuación, tenemos un enlace muy interesante que es una referencia completa, que nos permitirá consultar si ciertas propiedades o métodos que utilizamos en JavaScript, son compatibles en todos los navegadores: Tablas de compatibilidades entre navegadores

https://www.falconmasters.com/recursos-herramientas/compatibilidad-navegadores-can-i-use/

http://caniuse.com/

1.2 EVOLUCIÓN Y CARACTERÍSTICAS DE LOS NAVEGADORES WEB

- World Wide Web.
  - Está formada por un Conjunto de recursos interconectados:

    •Componentes Físicos de internet: *Hubs*, repetidores, puentes, pasarelas, …

    •Protocolos de comunicaciones: TCP, IP, HTTP, FTP, SMTP.

    •Sistema de nombres de dominio (DNS) para la búsqueda y recuperación de recursos.

- La Configuración arquitectónica más habitual se basa en el modelo *Cliente/Servidor:*
  - *Cliente* es un componente consumidor de servicios.
  - *Servidor* es un proceso proveedor de servicios.

- Un Navegador web o Explorador Web es:

  - El Componente software que se utiliza en el cliente y que permite acceder al contenido ofrecido por los servidores de Internet sin la necesidad de que el usuario instale un nuevo programa.

  - Una Aplicación, distribuida habitualmente como software libre, que permite a un usuario acceder (y, normalmente, visualizar) a un recurso publicado por un servidor Web a través de Internet y descrito mediante una dirección URL (*Universal Resource Locator*).

- Ejemplos de Navegadores web:

  - **Internet Explorer**. Es el navegador de Microsoft hasta que apareció el Windows 10 con **Microsoft Edge**.
  - **Mozilla Firefox**. Se trata de un navegador de código abierto multiplataforma de gran aceptación.
  - **Google Chrome**. Es el navegador de Google compilado a partir de componentes de código abierto.
  - **Opera**. El Opera Browser es uno de los mejores navegadores que existe en la actualidad (¿?).
  - **Safari**. Es el navegador por defecto de los sistemas de Apple, aunque se han desarrollado versiones para Windows.
  - **Dolphin Browser**. Para dispositivos móviles inteligentes (*smartphones y tablets*). Específico para el sistema operativo Android, fue uno de los primeros en incluir soporte para navegación multitáctil.

Comparativa de los navegadores más utilizados en 2019:

https://www.pcworld.es/mejores-productos/internet/mejores-navegadores-web-3672988/

https://www.testdevelocidad.es/2019/02/01/navegadores-mas-utilizados-opciones/

https://webtematica.com/los-mejores-navegadores-web

Comparativa de los navegadores más utilizados en 2022:

https://www.stackscale.com/es/blog/top-navegadores-caracteristicas-comparativa-estadisticas/

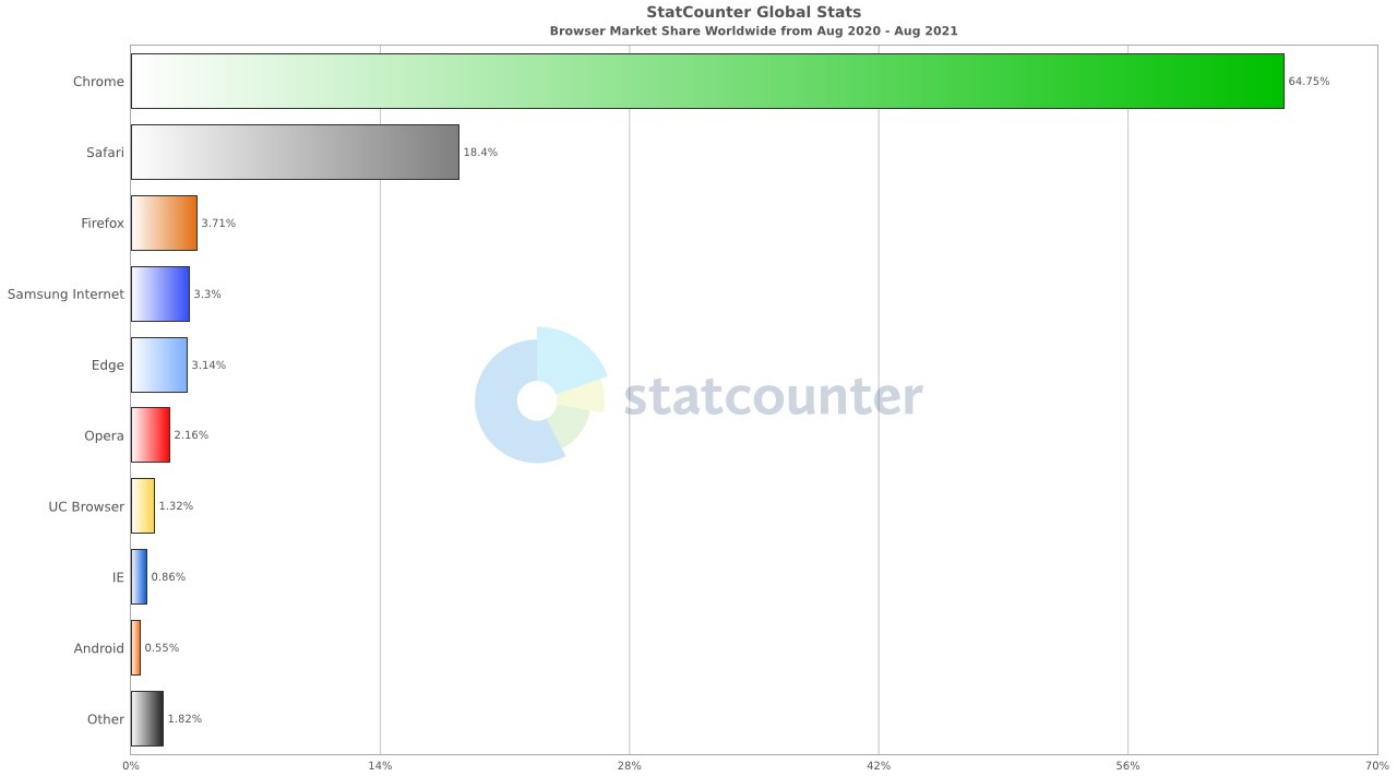
https://www.avast.com/es-es/c-fastest-web-browsers

Comparativa de los navegadores más utilizados en 2023:

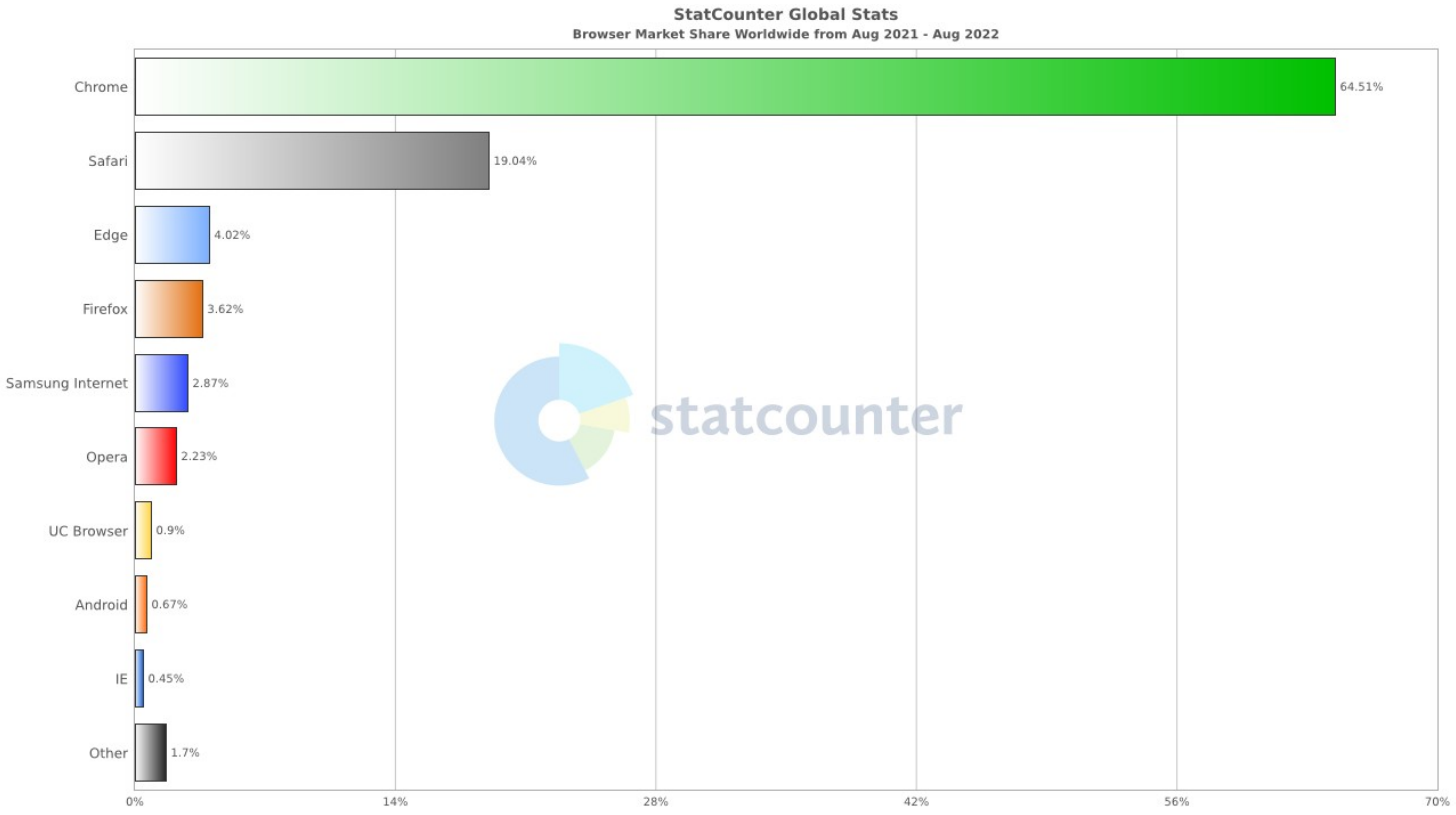
https://www.stackscale.com/es/blog/top-navegadores-caracteristicas-comparativa-estadisticas/

https://www.avast.com/es-es/c-fastest-web-browsers

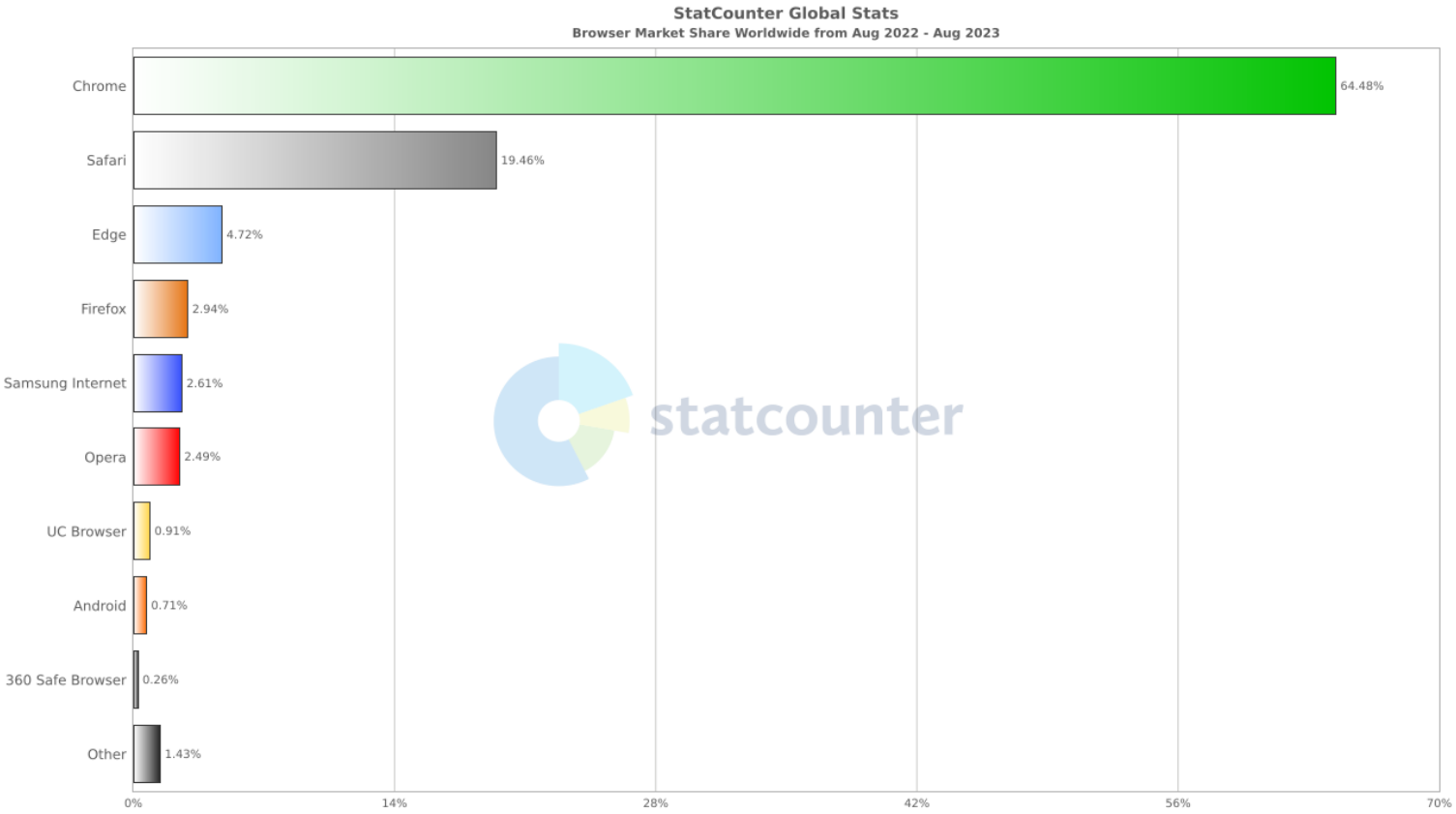
▪Estadísticas de uso de navegadores. Utilización de navegadores en el periodo **agosto 2020-agosto 2021** obtenida de ***gs.StatCounter.com*** es la siguiente:



▪Estadísticas de uso de navegadores. Utilización de navegadores en el periodo **agosto 2021-agosto 2022** obtenida de ***gs.StatCounter.com*** es la siguiente:



▪Estadísticas de uso de navegadores. Utilización de navegadores en el periodo **agosto 2022-agosto 2023** obtenida de ***gs.StatCounter.com*** es la siguiente:



Acceder a esta página para más información: **http://gs.statcounter.com/**

- Criterios de clasificación de los Navegadores web:

  - **Plataforma de ejecución**. Sistema operativo. No todos pueden ser ejecutados en cualquier Sistema Operativo. Safari es exclusivo de Apple.

  - **Características del navegador aportadas de forma nativa**. Funcionalidades adicionales como administración de marcadores, gestores de descarga, almacenamiento seguro de contraseñas y datos de formularios, etc. Todo esto sin necesidad de instalar extensiones.

  - **Personalización de la interfaz**. Funciones de accesibilidad: soporte para la navegación por pestañas, la existencia de bloqueadores de ventanas emergentes, visualización de ficheros en formato PDF, opciones de zoom, …

  - **Soporte de tecnologías Web**. Grado de soporte de los estándares de la Web como CSS, Java, lenguajes scripting del cliente (Javascript), XHTML (HTML con formato XML).

  - **Licencia de software**. Código libre como Mozilla Firefox o Google Chrome y navegadores propietarios como Internet Explorer/Microsoft Edge o Safari. Salvo raras excepciones (OmniWeb) todos los navegadores son gratuitos.

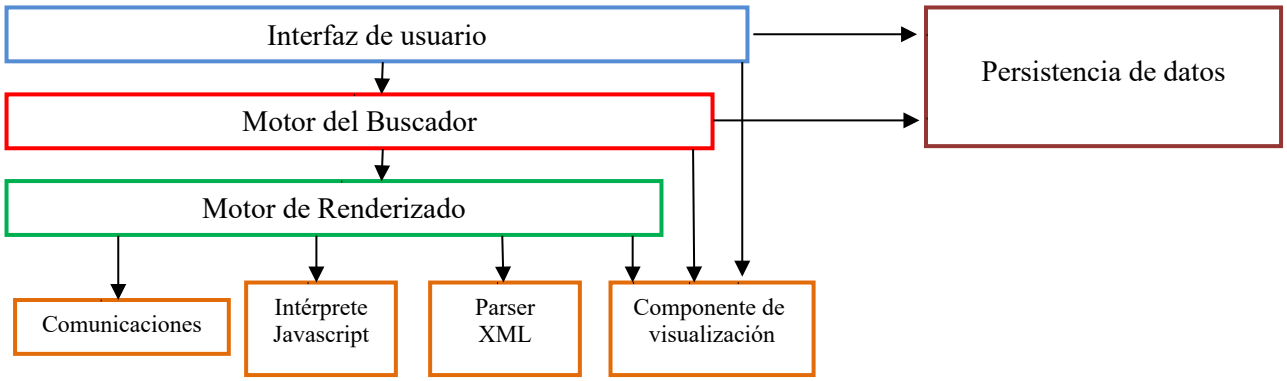
1.3 ARQUITECTURA DE EJECUCIÓN

▪Proceso de ejecución:

- Cada navegador web tiene su propia forma de interpretar la interacción con un usuario.
  Se inicia con el usuario indicando la dirección del recurso al que quiere acceder y termina con la visualización del recurso por parte del navegador en la pantalla del usuario.

▪Arquitectura de referencia de un navegador web:

Para poder llevar a cabo lo descrito anteriormente, cada navegador está formado por una serie de elementos y componentes determinados que conforman lo que se denomina **arquitectura del navegador**. De acuerdo con el estudio llevado a cabo por Grosskurth y Godfrey los componentes básicos incluidos en esta arquitectura son los siguientes:



- **Subsistema de interfaz de usuario**. Es la capa que actúa de interfaz entre el usuario y el motor del buscador (o de navegación). Ofrece funcionalidades tales como la visualización de barras de herramientas, progreso de carga de página, gestión inteligente de la descarga, preferencias de configuración de usuario e impresión.

- **Subsistema del motor del buscador o motor de navegación**. Este subsistema es un componente que ofrece una interfaz de alto nivel para el motor de renderizado. Su función principal es la de cargar una dirección determinada (URL o URI) y soportar los mecanismos básicos de navegación tales como ir página anterior o siguiente o la recarga de la página. Además, es el componente que gestiona las alertas de Javascript (mensajes de usuario). Por último, es el encargado de consultar y administrar las preferencias de ejecución del motor renderizado (significado de renderizado: proceso para generar una imagen).

- **Subsistema de renderizado**. Este componente es el encargado de producir una representación visual del recurso obtenido a partir del acceso a una dirección Web. El código de una página web es

interpretado por este módulo. Será capaz de mostrar documentos HTML, XML, hojas de estilo CSS e incluso contenido embebido en la página (audio/vídeo) e imágenes.

Algunos de los motores de renderizado más usados son:

- Gecko, utilizado en Firefox, Mozilla suite.
- Trident, motor de Internet Explorer para Windows
- WebKit, motor de Google Chrome, Safari, Epiphani
- Presto, el motor de Opera
- Tasman, motor de Internet Explorer para Mac
- Nuevo motor de renderizado Edge para el navegador Microsoft Edge (idéntico al WebKit)

○ **Subsistema de comunicaciones**. Es el subsistema encargado de implementar los protocolos de transferencia de ficheros y documentos utilizados en Internet (HTTP, FTP, etc.). Es el responsable de identificar la codificación de los datos obtenidos en función de su tipo: texto, audio, vídeo, etc.

○ **Intérprete de JavaScript**. Será el encargado de analizar y ejecutar código JavaScript.

○ **Parser XML**. Módulo que permite cargar en memoria una representación en árbol de la página web. Para poder acceder con más facilidad a los contenidos de un documento XHTML, los navegadores incluyen un módulo (parser) que permite cargar en memoria una representación en árbol (árbol DOM, Document Object Model) de la página. Con ello el acceso a los elementos de una página por parte del navegador es mucho más rápido.

○ **Componente de visualización**. Este subsistema ofrece funcionalidades relacionadas con la visualización de los contenidos de un documento HTML en una página web. Ofrece primitivas de dibujo y posicionamiento en una ventana, un conjunto de componentes visuales predefinidos (*widgets*) y un conjunto de fuentes tipográficas a los subsistemas principales del navegador web.

○ **Subsistema de persistencia de datos**. Funciona como almacén de diferentes tipos de datos para los principales subsistemas del navegador. Como historiales de navegación y el mantenimiento de sesiones de usuario en disco.

1.4 LENGUAJES Y TECNOLOGÍAS DE PROGRAMACIÓN EN ENTORNO CLIENTE

Los lenguajes de programación del entorno de cliente son aquellos que se ejecutan en el navegador Web, o sea, en el lado del cliente en una arquitectura cliente/servidor.

El lenguaje cliente principal para definir la estructura y el contenido de la página en forma de texto es el HTML (HyperText Markup Language, lenguaje de marcado de hipertexto). Existen alternativas como XML (lenguaje de marcas extensible, eXtensible Markup Language), DHTML (Dynamic HTML) o XHTML (eXtensible HTML).

Con el fin de mejorar la interactividad con el usuario, se incluyen todos los lenguajes script: JavaScript (el más utilizado), VBScript.

Lenguajes de programación más populares 2019:

http://ceeivalencia.emprenemjunts.es/?op=8&n=18532

https://www.locurainformaticadigital.com/2019/03/28/lenguajes-de-programacion-mas-demandados-2019/

https://blog.educacionit.com/2019/06/21/estos-son-los-lenguajes-de-programacion-mas-populares-segun-la-cantidad-de-tutoriales-que-se-buscan-en-google/

Indice Tiobe 2019: Hay muchos parámetros para medir la popularidad de los lenguajes de programación. TIOBE es un índice que busca saber qué lenguajes están en boga.

https://www.unocero.com/software/lenguajes-programacion-mas-populares-indice-tiobe/

https://fyaromo.com.co/2019/04/14/los-lenguajes-de-programacion-mas-populares-indice-tiobe-abril-de-2019/

Lenguajes de programación más populares 2020:

https://www.tokioschool.com/noticias/lenguajes-de-programacion-2020-esenciales-para-este-ano/

https://es.classpert.com/blog/lenguajes-de-programacion-mas-usados

https://www.tuexperto.com/2020/03/11/los-10-lenguajes-de-programacion-con-mas-futuro-en-2020/

Lenguajes de programación más populares 2021:

https://diariodigitalis.com/programacion/2021/05/05/indice-tiobe-de-mayo-python-adelanta-a-java-una-vez-mas/

https://www.vbote.com/actualidad-detalle/dynacontent/lenguajes-de-programacion-que-debes-conocer-en-2021.html

https://www.digitalhouse.com/ar/blog/top-5:-los-lenguajes-de-programacion-mas-utilizados-en-2021

Lenguajes de programación más populares 2022:

https://keepcoding.io/blog/5-lenguajes-de-programacion-mas-usados-2022/

Indice TIOBE _ febrero 2022: https://www.itmastersmag.com/noticias-analisis/12-lenguajes-de-programacion-con-demanda-en-2022/

https://www.tiobe.com/tiobe-index/

Lenguajes de programación más populares 2023:

https://keepcoding.io/blog/5-lenguajes-de-programacion-mas-usados-2022/

Indice TIOBE _ febrero 2022: https://www.itmastersmag.com/noticias-analisis/12-lenguajes-de-programacion-con-demanda-en-2022/

https://www.tiobe.com/tiobe-index/

1.5 INTEGRACIÓN DEL CÓDIGO CON LAS ETIQUETAS HTML

Para añadir javascript a nuestra página web necesitamos:

Un editor de texto: bloc de notas, notepad++, ultraedit, …
Un navegador: Mozilla Firefox, Google Chrome, Internet Explorer, …

RECORDAR:

- La extensión de los archivos que contengan X(HTML) y JavaScript serán .htm o .html
- En el caso de archivo externo con código javascript la extensión será .js
- Los archivos X(html) aporta la estructura, CSS el estilo y javascript el comportamiento.
- Para añadir cualquier comentario en (X)html:
  <!-- Esto es un comentario -->

La integración del código Javascript con HTML/XHTML podemos hacerla de tres formas diferentes: Javascript en el mismo documento, en un archivo externo o en elementos de HTML.

1.5.1 JavaScript en el mismo documento HTML

- Uso de unas etiquetas predefinidas para marcar el texto (<script> y </script>).
- Puede incluirse en cualquier parte del documento, aunque se recomienda que se defina dentro de la cabecera del documento HTML, entre <head> y </head>.
- Esta técnica suele utilizarse cuando se definen instrucciones que se referenciarán desde cualquier parte del documento o cuando se definen funciones con fragmentos de código genéricos.

```
<!DOCTYPE html>

<html lang="es">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
     <meta name="Keywords" content="Html5, css3, Javascript">

     <title>Ejemplo 1</title>
     <script type="text/javascript">
          alert("Prueba de JavaScript");
     </script>
  </head>
  <body>
     <h1>Ejemplo 1: código embebido</h1>
  </body>
</html>
```

1.5.2 JavaScript en un archivo externo

- Las mismas instrucciones de JavaScript que se incluyen entre un bloque <script></script> pueden almacenarse en un fichero externo con extensión .js.
- La forma de acceder y enlazar esos ficheros .js con el documento HTML/XHTML es a través de la propia etiqueta <script>.
- No existe un límite en el número de ficheros .js que pueden enlazarse en un mismo documento HTML/ XHTML.

Archivo **mensaje.js** contendría:

```
alert("Prueba de JavaScript");
```

Archivo **ejemplo2.html**:

```
<!DOCTYPE html>

<html lang="es">
   <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Ejemplo 2</title>
    <script type="text/javascript" src="/inc/mensaje.js"></script>
   </head>
   <body>
          <h1>Ejemplo 2: fichero externo</h1>
   </body>
</html>
```

1.5.3 JavaScript en elementos HTML (como respuesta a eventos)

- Consiste en insertar fragmentos de JavaScript dentro de atributos de etiquetas HTML de la página.
- Forma de controlar los eventos que suceden asociados a un elemento HTML concreto.
- Principal desventaja: el mantenimiento y modificación del código puede resultar más complicado.

```
<!DOCTYPE html>

<html lang="es">
   <head>
     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" /
>
     <title>Ejemplo 3</title>
   </head>
   <body>
     <p onclick="alert('Prueba de JavaScript');”>
     Ejemplo 3: código en atributos
     </p>
   </body>
</html>
```

Ejemplo anterior: Respuesta_a_eventos.html evento onclick() con <button>, con <a href= > y con <p>

OTROS EJEMPLOS:

Ejemplo 1: con **script situado en la cabecera <head>**, utilizamos la función (declarada por el programador) yetAnotherAlert con el argumento textToAlert, la misión de esta función es que aparezca una alerta en la ventana del navegador con el texto especificado en el argumento de la función alert(textToAlert); (alert() es una función predeterminada por el lenguaje)

El script llama a la función declarada con un argumento entrecomillado entre paréntesis yetAnotherAlert("This is Chapter 2");

```
<script type = "text/javascript">
        function yetAnotherAlert(textToAlert) {
                alert(textToAlert);
                }
        yetAnotherAlert("This is Chapter 2");
 </script>
```

eclipse_myfirstpage.htm

EJERCICIO ALUMNADO: OJO probar colocando la función en <body>,1º colocar la llamada y a continuación la función en el mismo script. 2º colocar en un primer script la llamada y luego en otro script la función. ¿Cuál es la diferencia? ¿Por qué?

Ejemplo 2: el mismo ejemplo anterior pero con **el script definido en un archivo externo al X(html)**, la editamos y lo guardamos con extensión .js, y en este caso el archivo se llamará myscript.js este archivo contendrá:

```
function yetAnotherAlert(textToAlert) {
                alert(textToAlert);
                }
yetAnotherAlert("This is the Second Example.");
```

En nuestro archivo .html, en <head> pondremos:

```
<script  type = "text/javascript" src=”myscript.js”> </script>
```

myfirstpage.htm myscript.js

Ejemplo 3: sencillo usando función alert() en <body>:

Bien para poner mensajes se utiliza mucho la función alert(“”). Por ejemplo alert(“Hola”);

```
<html>
<head>
<title> Mi Primer JavaScript </title>
</head>
<body>
	<hr />
	<script language="JavaScript">
		alert (“Hola Mundo en JavaScript”);
	</script>
	<hr />
</body>
</html>
```

hola mundo.html