UD1

Selección de arquitectura y herramientas de programación

1. Evolución y características de los navegadores web

En los inicios de Internet había bastantes problemas de compatibilidad entre navegadores (Internet Explorer, Netscape Navigator, Opera) a diferentes niveles, y esto hizo tomar conciencia de la importancia en la adopción de **estándares**. El **World Wide Web Consortium** (WWWC o W3C) es una organización que se encarga de velar por la **implementación de estándares en el ámbito web**, haciendo posible que diferentes fabricantes de software se pongan de acuerdo en beneficio del usuario final.

1.1. Arquitectura Cliente/Servidor:

En entornos web, la configuración arquitectónica más habitual se basa en el modelo denominado **Cliente/Servidor**, basado en la idea de **servicio**, en el que el **cliente** es un componente **consumidor de servicios** y el **servidor** es un proceso **proveedor de servicios**. Además, esta relación está robustamente cimentada en el **intercambio de mensajes** como el único elemento de acoplamiento entre ambos.

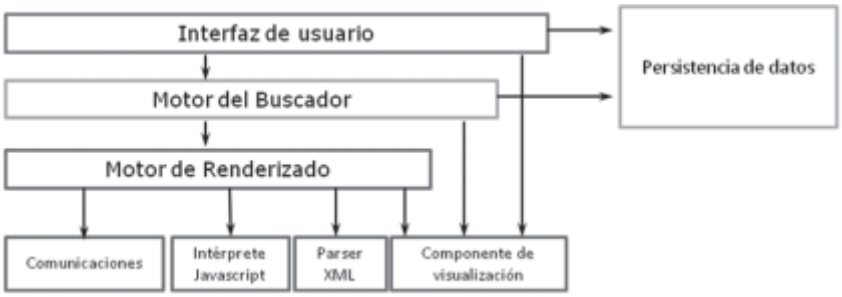
1.2. Definición de navegador web:

Un navegador web, o explorador web (browser), es 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)**.

Desde la creación de la Web a principios de los años 90, los navegadores web han evolucionado desde meros visualizadores de texto que, aunque no ofrecían capacidades multimedia (visualización de imágenes), cumplían su propósito (Links, Lynx, W3M); hasta los actuales navegadores, totalmente preparados para soportar cualquier tipo de interacción y funcionalidad requerida por el usuario. Algunos de los navegadores actuales más extendidos son Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera, etc.

1.3. Arquitectura de ejecución de navegadores web:

Para poder llevar a cabo el proceso de navegación web, cada navegador está formado por una serie de elementos y componentes determinados que conforman lo que se denomina **arquitectura del navegador**. A pesar de que cada navegador tiene su propia arquitectura, la gran mayoría de ellos coinciden en una serie de componentes básicos y comunes en todos ellos, es lo que llamamos arquitectura de referencia. Los componentes básicos incluidos en la arquitectura de referencia de un navegador web son los que pueden verse en la siguiente figura:



Los componentes de esta arquitectura de referencia son:

- **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 la página, gestión inteligente de las descargas, preferencias de configuración de usuario o impresión.

- **Subsistema del motor del buscador o motor de navegación**. Su función principal es la de cargar una dirección determinada (URL o URI) y soportar los mecanismos básicos de navegación tales como ir a la página anterior o siguiente o la recarga de la página. Además, es el componente que gestiona el proceso de carga de una página (es quien le provee de información a la interfaz de usuario al respecto).

- **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. En función de los lenguajes, estándares y tecnologías soportadas por

el navegador, este módulo será capaz de mostrar documentos HTML, XML, hojas de estilo CSS e incluso contenido embebido en la página (audio/vídeo) e imágenes. Además, este módulo establece las dimensiones exactas de cada elemento a mostrar y, en ocasiones, es el responsable de posicionar dichos elementos en una página.

Algunos de los motores de renderizado más utilizados son:

– **Gecko**, utilizado en Firefox, Mozilla Suite y otros navegadores como Galeon.

– **Trident**, el motor de Internet Explorer para Windows.

– **WebKit**, el motor de Google Chrome (en sus inicios), Epiphany y Safari.

– **Presto**, el motor de Opera.

– **Tasman**, el motor de Internet Explorer para Mac.

– **Blink**, es el nuevo motor de renderizado de Google Chrome.

- **Subsistema de comunicaciones**: Es el subsistema encargado de implementar los protocolos de transferencia de ficheros y documentos utilizados en Internet (HTTP, FTP, etc.).

- **Intérprete o motor de JavaScript**: Las páginas HTML habitualmente llevan código intercalado para la provisión de ciertas funcionalidades al usuario como puede ser la respuesta a ciertos eventos del ratón o del teclado. El lenguaje comúnmente aceptado para la programación de este código embebido es JavaScript (siguiendo el estándar ECMAScript). El intérprete de JavaScript será el encargado de analizar y ejecutar dicho código. Los motores JavaScript **son exclusivos de cada navegador**, y constituyen un **elemento fundamental para la velocidad** a la que cada browser es capaz de interpretar las instrucciones y realizar el renderizado de la página a cargar.

  → **Nota:** En definitiva, la combinación **"Layout Engine" + "Javascript Engine"** es la que determina la **velocidad** a la que cada navegador carga las páginas web.

  Los JavaScript Engines adquirieron importancia a raíz de la aparición de Google Chrome en 2008, que literalmente barrió a la competencia en velocidad de ejecución. Pronto Mozilla y WebKit (Safari) reaccionaron, entablándose una auténtica carrera.

  Los motores JavaScript evolucionan tan rápido que es difícil seguir su actual estado de desarrollo. Dentro de los motores más extendidos tenemos:

  - **V8 Engine**, creado por **Google**, es open source y utilizado por el navegador **Google Chrome**. Además de funcionar en este famoso navegador, también lo han adaptado para correr de lado

servidor, haciendo que JavaScript sea utilizado como **lenguaje de programación back-end** (Node JS).

Repositorio Oficial: https://github.com/v8/v8

- **Chakra**, creado por **Microsoft,** es utilizado por su navegador web **Internet Explorer** 9 y en su nuevo navegador web **Microsoft Edge** (incluido en sus Sistemas Operativos Windows 10). En el 2015, liberaron el código fuente de su motor JScript convirtiéndolo en open source.

  Repositorio Oficial: https://github.com/Microsoft/ChakraCore

- **SpiderMonkey**, creado por la **Fundación Mozilla**, es utilizado por su navegador web **Mozilla Firefox.**

  Repositorio Oficial: SpiderMonkey Project.

- **Carakan**, creado por **Opera Software**, utilizado por su navegador **Opera.**

- **Apple** evoluciona su misterioso motor **Nitro** para su navegador **Safari**.

- **Parser XML/JSON:** los navegadores web suelen incluir un módulo (parser) que permite cargar en memoria una representación en árbol **(árbol DOM, Document Object Model)** de la página. De esta forma, el acceso a los diferentes elementos de una página por parte del navegador es mucho más rápido.

- **Componente de Visualización:** 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. Suele estar muy relacionado con las librerías de visualización del sistema operativo.

- **Subsistema de persistencia de datos**: Funciona como almacén de diferentes tipos de datos para los principales subsistemas del navegador. Estos datos suelen estar relacionados con el almacenamiento de historiales de navegación y el mantenimiento de sesiones de usuario en disco.

2. 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, dicho de otro modo, en el lado del cliente dentro de una arquitectura Cliente/Servidor.

El lenguaje cliente principal es **HTML** (lenguaje de marcado de hipertexto, HyperText Markup Language), ya que la mayoría de páginas del servidor son codificadas siguiendo este lenguaje para describir la estructura y el contenido de una página en forma de texto. Existen algunas alternativas y variaciones de este lenguaje tales 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, en este grupo de lenguajes cliente podemos incluir todos los lenguajes de script, tales como **JavaScript** (el más utilizado dentro de esta rama) o **VBScript**. También existen otros lenguajes más independientes, como **ActionScript** (para crear contenido Flash) o **AJAX** (como extensión a JavaScript para comunicación asíncrona).

2.1. JavaScript.

JavaScript es un lenguaje de programación interpretado, dialecto del estándar **ECMAScript**. Se define como **orientado a objetos**, **interpretado**, **imperativo**, **débilmente tipado** y **dinámico**.

Se utiliza principalmente en el lado del cliente, implementado como parte de un navegador web permitiendo crear interacción con el usuario y páginas web dinámicas, aunque actualmente es posible ejecutar JavaScript en el propio servidor (NodeJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF o aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

JavaScript se diseñó con una sintaxis similar al lenguaje de programación C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

2.2. Breve historia.

A principios de los años 90, la mayoría de usuarios que se conectaban a Internet lo hacían con módems a una velocidad máxima de 28.8 kbps. En esa época, empezaban a desarrollarse las primeras aplicaciones web y por tanto, las páginas web **comenzaban a incluir formularios complejos**.

Con unas aplicaciones web cada vez más complejas y una velocidad de navegación tan lenta, surgió la **necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario**. De esta forma, si el usuario no rellenaba correctamente un formulario, no se le hacía esperar mucho tiempo hasta que el servidor volviera a mostrar el formulario indicando los errores existentes.

Brendan Eich, un programador que trabajaba en **Netscape**, pensó que podría solucionar este problema adaptando otras tecnologías existentes (como ScriptEase) al navegador Netscape Navigator 2.0, que iba a lanzarse en 1995. Inicialmente, Eich denominó a su lenguaje LiveScript.

Posteriormente, **Netscape firmó una alianza con Sun Microsystems** para el desarrollo del nuevo lenguaje de programación. Además, justo antes del lanzamiento Netscape decidió cambiar el nombre por el de JavaScript. La razón del cambio de nombre fue exclusivamente por **marketing**, ya que Java era la palabra de moda en el mundo informático y de Internet de la época.

La primera versión de JavaScript fue un completo éxito y Netscape Navigator 3.0 ya incorporaba la siguiente versión del lenguaje, la versión 1.1. Al mismo tiempo, **Microsoft** lanzó **JScript** con su navegador Internet Explorer 3. JScript era una copia de JavaScript al que le cambiaron el nombre para evitar problemas legales.

Para evitar una guerra de tecnologías, Netscape decidió que lo mejor sería estandarizar el lenguaje JavaScript. De esta forma, en 1997 se envió la especificación JavaScript 1.1 al organismo **ECMA (**European Computer Manufacturers Association).

ECMA creó el comité TC39 con el objetivo de "estandarizar de un lenguaje de script multiplataforma e independiente de cualquier empresa". El primer estándar que creó el comité TC39 se denominó **ECMA-262**, en el que se definió por primera vez el lenguaje ECMAScript.

Por este motivo, **algunos programadores prefieren la denominación ECMAScript para referirse al lenguaje JavaScript**. De hecho, JavaScript no es más que la implementación que realizó la empresa Netscape del estándar ECMAScript.

La organización internacional para la estandarización (ISO) adoptó el estándar ECMA-262 a través de su comisión IEC, **dando lugar al estándar ISO/IEC-16262**.

2.3 Especificaciones oficiales

ECMA ha publicado varios estándares relacionados con ECMAScript. En Junio de 1997 se publicó la primera edición del estándar ECMA-262. Un año después, en Junio de 1998 se realizaron pequeñas modificaciones para adaptarlo al estándar ISO/IEC-16262 y se creó la segunda edición.

La novena edición del estándar **ECMA-262** (publicada en junio de 2018) es la versión que utilizan los navegadores actuales (gradualmente incorporado) y se puede consultar gratuitamente en

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

2.4 Cómo incluir JavaScript en documentos HTML

La integración de JavaScript y HTML es muy flexible, ya que existen al menos 3 formas para incluir código JavaScript en las páginas web.

2.4.1 Incluir JavaScript en el mismo documento HTML

El código JavaScript se encierra entre etiquetas **<script>** y se incluye en cualquier parte del documento. Aunque es correcto incluir cualquier bloque de código en cualquier zona de la página, se recomienda definir el código JavaScript dentro de la cabecera del documento (dentro de la etiqueta <head>):

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Ejemplo de código JavaScript en el propio documento</title>
    <script type="text/javascript">
      console.log("Un mensaje de prueba");
    </script>
</head>
<body>
    <p>Un párrafo de texto.</p>
</body>
</html>
```

Para que la página HTML resultante sea válida, es necesario añadir el atributo **type** a la etiqueta **<script>**. Los valores que se incluyen en el atributo type están estandarizados y para el caso de JavaScript, el valor correcto es **text/javascript**.

Este método se emplea cuando se define un bloque pequeño de código o cuando se quieren incluir instrucciones específicas en un determinado documento HTML que completen las instrucciones y funciones que se incluyen por defecto en todos los documentos del sitio web.

El **principal inconveniente** es que si se quiere hacer una modificación en el bloque de código, es necesario modificar todas las páginas que incluyen ese mismo bloque de código JavaScript.

2.4.2 Definir JavaScript en un archivo externo

Las instrucciones JavaScript se pueden incluir en un archivo externo de tipo JavaScript que los documentos HTML enlazan mediante la etiqueta **<script>**. Se pueden crear todos los archivos JavaScript que sean necesarios y cada documento HTML puede enlazar tantos archivos JavaScript como necesite.

Ejemplo:

Archivo codigo.js

```
console.log("Un mensaje de prueba");
```

Documento HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Ejemplo de código JavaScript en el propio documento</title>
    <script type="text/javascript" src="/js/codigo.js"></script>
```

```
</head>
<body>
    <p>Un párrafo de texto.</p>
</body>
</html>
```

Además del atributo **type**, este método requiere definir el atributo **src**, que es el que indica la URL correspondiente al archivo JavaScript que se quiere enlazar. Cada etiqueta **<script>** solamente puede enlazar un único archivo, pero en una misma página se pueden incluir tantas etiquetas **<script>** como sean necesarias.

Los archivos de tipo JavaScript son documentos normales de texto con la **extensión .js,** que se pueden crear con cualquier editor de texto como Notepad, Wordpad, EmEditor, UltraEdit, Vi, etc.

La **principal ventaja** de enlazar un archivo JavaScript externo es que se simplifica el código HTML de la página, que se puede reutilizar el mismo código JavaScript en todas las páginas del sitio web y que cualquier modificación realizada en el archivo JavaScript se ve reflejada inmediatamente en todas las páginas HTML que lo enlazan.

2.4.3 Incluir JavaScript en los elementos HTML

Este último método es el **menos utilizado**, ya que consiste en incluir instrucciones JavaScript dentro del código HTML de la página:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Ejemplo de código JavaScript en el propio documento</title>
</head>
<body>
    <p onclick="console.log('Un mensaje de prueba')">Un párrafo de texto.</p>
</body>
</html>
```

El mayor inconveniente de este método es que **ensucia innecesariamente el código HTML** de la página y **complica el mantenimiento** del código JavaScript. En general, este método sólo se utiliza para definir algunos eventos y en algunos otros casos especiales, como se verá más adelante.

2.5 Etiqueta noscript

Algunos navegadores no disponen de soporte completo de JavaScript, otros navegadores permiten bloquearlo parcialmente e incluso algunos usuarios bloquean completamente el uso de JavaScript porque creen que así navegan de forma más segura.

En estos casos, es habitual que si la página web requiere JavaScript para su correcto funcionamiento, se incluya un mensaje de aviso al usuario indicando que debería activar JavaScript para disfrutar completamente

de la página. El siguiente ejemplo muestra una página web basada en JavaScript cuando se accede con JavaScript activado y cuando se accede con JavaScript completamente desactivado.

El lenguaje HTML define la etiqueta **<noscript>** para mostrar un mensaje al usuario cuando su navegador no puede ejecutar JavaScript. El siguiente código muestra un ejemplo del uso de la etiqueta <noscript>:

```
<head> ... </head>
<body>
    <noscript>
      <p>Bienvenido a Mi Sitio</p>
      <p>La página que estás viendo requiere para su funcionamiento el
      uso de JavaScript. Si lo has deshabilitado intencionadamente,
      por favor vuelve a activarlo.</p>
    </noscript>
</body>
```

La etiqueta **<noscript>** se debe incluir en el interior de la etiqueta **<body>** (normalmente se incluye al principio de <body>). El mensaje que muestra <noscript> puede incluir cualquier elemento o etiqueta XHTML.

2.6. AJAX

AJAX, acrónimo de **Asynchronous JavaScript And XML** (JavaScript Asíncrono y XML), es un conjunto de técnicas y métodos de desarrollo web para la creación aplicaciones web interactivas. El **primer aspecto** que define a AJAX es que este tipo de aplicaciones se **ejecutan en el cliente**, es decir, en el navegador de los usuarios que acceden a una página web. La **segunda característica** es que, al contrario que con una página web HTML (con o sin JavaScript), en la que la comunicación se interrumpe una vez el cliente recibe la página, **con AJAX se mantiene una comunicación asíncrona con el servidor en segundo plano** (sin que el usuario sea consciente de dicha comunicación). La consecuencia directa de esta técnica es que **podemos realizar cambios sobre las páginas del cliente sin necesidad de que éste proceda a recargarlas**. Este hecho implica un aumento de la interactividad con el usuario y de la velocidad en las aplicaciones.

El **fundamento de AJAX** se encuentra en la utilización de un objeto específico de JavaScript denominado **XMLHttpRequest**, disponible y aceptado por la mayoría de los navegadores actuales.

3. Inspector de código en los navegadores:

El inspector de código es una **herramienta para desarrolladores que viene integrada en los navegadores modernos** como Firefox y Google Chrome. Tienes varias formas de abrir el inspector de código, una de mis favoritas es con el acceso por teclas.

- *En Mac y Linux*: **Cmd + Opción + i**
- *En Windows* : **Ctrl + Shift + i**