Desarrollo Web en entorno cliente

UT 1. Introducción

T.S. en Desarrollo de Aplicaciones Web

IES Escultor J.L. Sánchez

Contenido

1. La web

La WEB es más que uno de los muchos servicios que presta Internet, entre los que podemos encontrar, entre otros:

- Correo electrónico
- FTP
- La propia web

El sistema con el que está construida la web se llama **hipertexto** y es un entramado de páginas conectadas con enlaces.

La web no sólo se limita a presentar textos y enlaces, sino que también puede ofrecernos imágenes, videos, sonido y todo tipo de presentaciones, llegando a ser el servicio más rico en medios que tiene Internet. Por esta razón, para referirnos al sistema que implementa la web (hipertexto), se ha acuñado el término que hipermedia, haciendo referencia a que la web permite **contenidos multimedia**.

Una página web que vemos en nuestro navegador, o cliente web, está compuesta por multitud de diferentes componentes: HTTML, elementos multimedia, hipervínculos y/o script.

El lenguaje de diseño de páginas web es **HTML**; es un lenguaje de marcado que indica básicamente dónde colocar cada texto, cada imagen o cada video y la forma que tendrán éstos al ser colocados en la página.

En la web podemos encontrar, o construir, dos tipos de páginas:

- **Estáticas**: las que se presentan sin movimiento y sin funcionalidades más allá de los enlaces; se construyen con HTML.
- **Dinámicas**: las páginas que tienen efectos especiales y en las que podemos interactuar; es necesario utilizar otros lenguajes además de HTML

Tipos de página dinámicas:

- **Páginas dinámicas del lado del cliente:** se procesan en el cliente; toda la carga de procesamiento la soporta el navegador web. Usos típicos de las páginas de cliente son efectos especiales para webs como objetos animados, control de formularios, cálculos, etc. El código necesario para crear los efectos y funcionalidades se incluye dentro del mismo archivo HTML y es llamado **SCRIPT**. Cuando una página HTML contiene scripts de cliente, el navegador se encarga de interpretarlos y ejecutarlos para realizar los

efectos y funcionalidades. El lenguaje utilizado para diseñar script es, principalmente, **Javascript**.

- **Páginas dinámicas del lado del servidor**: son reconocidas, interpretadas y ejecutadas por el propio servidor. Con ellas se puede hacer todo tipo de aplicaciones web, desde agendas a foros, sistemas de documentación, estadísticas, juegos, chats, etc.

  Son especialmente útiles tareas en las que se tiene que acceder a información centralizada, situada en una base de datos en el servidor, y cuando por razones de seguridad los cálculos no se pueden realizar en el ordenador del usuario.

  En las páginas dinámicas del lado del servidor, el código fuente también se puede incluir, según el lenguaje utilizado, en el propio archivo HTML, al igual que ocurre en las páginas del cliente. Cuando una página es solicitada por parte de un cliente, el servidor ejecuta los scripts y se genera una página resultado, que sólamente contiene código HTML. Este resultado final es el que se envía al cliente y puede ser interpretado sin lugar a errores ni incompatibilidades, puesto que sólo contiene HTML.

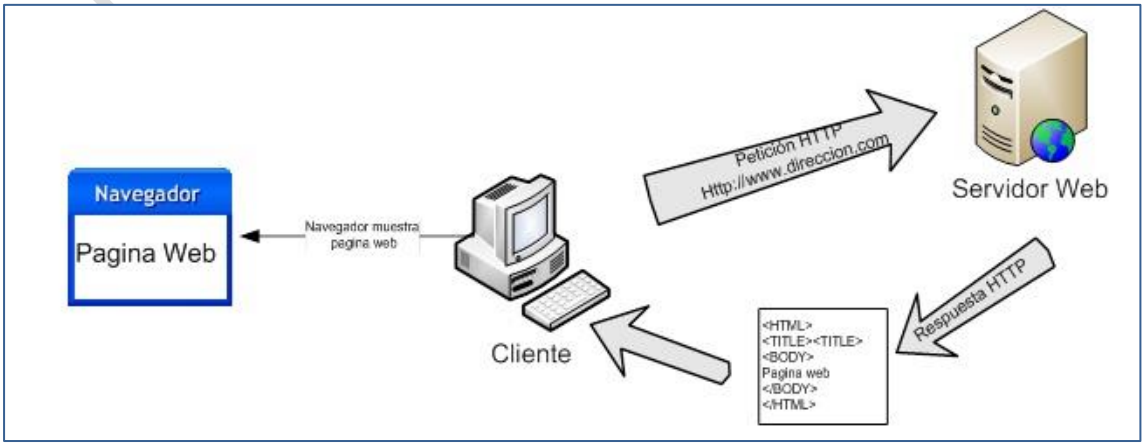
  Lenguajes del lado del servidor: *PHP*, *ASP*, *JSP*, ...

2. Arquitectura cliente-servidor

La **arquitectura** de un sitio web tiene tres componentes principales:

- Un servidor Web
- Una conexión de red
- Los clientes

El servidor Web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red, y para ello se usa el protocolo HTTP. Una vez que se solicita esta petición mediante el protocolo HTTP y la recibe el servidor Web, éste localiza el recurso solicitado en su sistema de archivos y envía la respuesta adecuada de vuelta al navegador que la solicitó.



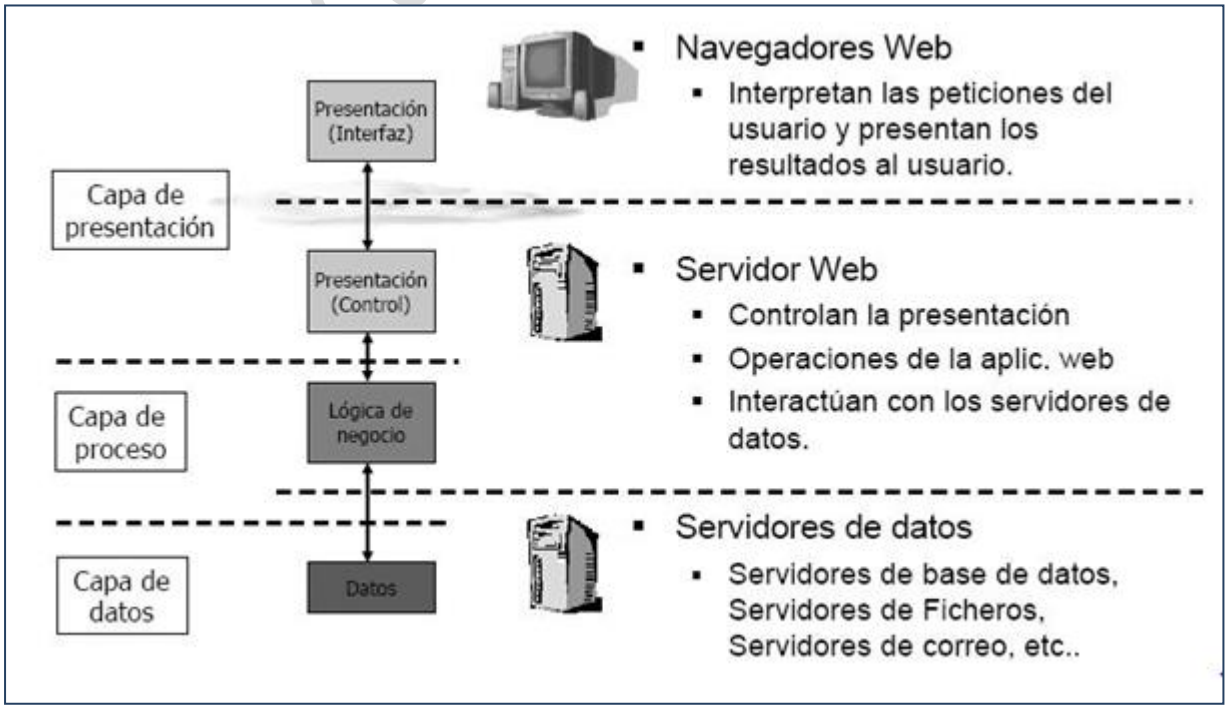
Las **aplicaciones web** están basadas en el modelo **Cliente/Servidor** que gestionan servidores web, y que utilizan como interfaz páginas web.

Las páginas Web son el componente principal de una aplicación o sitio Web. Los browsers piden páginas (almacenadas o creadas dinámicamente) con información a los servidores Web. Una vez que se entrega una página al cliente, la conexión entre el navegador y el servidor Web se rompe.

Las aplicaciones web se modelan mediante lo que se conoce como **modelo de capas**. Una capa representa un elemento que procesa o trata información.

El modelo más utilizado es el **modelo de tres capas**.

1. **Capa de presentación** (parte en el cliente y parte en el servidor)
   - Recoge la información del usuario y la envía al servidor (cliente)
   - Manda información a la capa de proceso para su procesado
   - Recibe los resultados de la capa de proceso
   - Generan la presentación
   - Visualizan la presentación al usuario (cliente)
2. **Capa de proceso** (servidor web)
   - Recibe la entrada de datos de la capa de presentación
   - Interactúa con la capa de datos para realizar operaciones
   - Manda los resultados procesados a la capa de presentación
3. **Capa de datos** (servidor de datos)
   - Almacena los datos, asegurando la integridad de los mismos
   - Recupera y actualiza los datos



3. Lenguajes del lado del cliente

Los lenguajes de programación del lado cliente se integran en las páginas web. Un código escrito en un lenguaje de **script** se incorpora directamente dentro de un código HTML y se ejecuta **interpretado.**

Las páginas creadas con estos lenguajes son enviadas al usuario, de forma que el navegador es el encargado de interpretar y mostrarlas por pantalla. Al ser tecnología de tipo cliente, habrá que prestar especial atención al navegador utilizado, ya que, como ocurre en muchas ocasiones, lo que funciona con uno puede no hacerlo con otro, aunque todos dicen estar basados en los estándares del **W3C Consortium**: Comité de generación de estándares para la *Word Wide Web*, similar a la *ISO*, que se encarga de generar versiones estándares de HTML, SGML(Standard Generalized Mark-up Lenguaje), XML (Extensible MarkupLanguage), XHTML, las plantillas de estilos CSS (Cascade Style Sheets) etc.

La principal **ventaja** de los lenguajes del lado del cliente es que son totalmente independientes del servidor, lo que permite que la página pueda ser albergada en cualquier sitio.
Entre los **inconvenientes** de estos lenguajes:

- nuestra página no se visualizará o funcionará correctamente si el navegador no tiene instalados/activos los componentes necesarios para ejecutar el script
- el código de la página web, incluido el código de los scripts, son accesibles a cualquiera y, en consecuencia, ello puede afectar a su seguridad.

En el tutorial de Javascript de *w3schools* se puede consultar la compatibilidad de cada elemento del lenguaje en cada uno de los navegadores actuales:
https://www.w3schools.com/js/default.asp

4. Introducción a la programación en Javascript

Javascript es un lenguaje de programación que surgió con el objetivo inicial de programar ciertos comportamientos sobre las páginas web, respondiendo a la interacción del usuario y la realización de automatismos sencillos. En ese contexto podríamos decir que nació como

un "lenguaje de scripting" del lado del cliente, sin embargo, hoy Javascript es mucho más. Las necesidades de las aplicaciones web modernas y HTML5 ha provocado que el uso de Javascript que encontramos hoy haya llegado a unos niveles de complejidad y prestaciones tan grandes como otros lenguajes de primer nivel.

Javascrip fue diseñado por Brendan Eich en 1995 para el navegador NetScape; fue diseñado ¡en 10 días! Inicialmente se llamó *Mocha*, después pasó a llamarse *LiveScript* y finalmente tomó el nombre de Javascript.
Entre 1996 y 1997 se diseñó un estándar para Javascript llamado **ECMAScript** para intentar dar compatibilidad al lenguaje con los distintos navegadores.

Con Javascript podemos crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones Javascript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso con que cuenta este lenguaje es el propio navegador y todos los elementos que hay dentro de una página. Actualmente, gracias a las API Javascript de HTML5, que están disponibles en los navegadores actuales de ordenadores y dispositivos, podemos acceder a todo tipo de recursos adicionales, como la cámara, espacio para almacenamiento de datos, creación de gráficos basados en vectores y mapas de bits, flujos de datos con servidores, etc. Con todo ello se han multiplicado las posibilidades.
Javascript es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, utilizando objetos, funciones, estructuras de datos complejas, etc. Además, Javascript pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente.

Para el desarrollo de sitios web sería suficiente generalmente con el uso de Javascript puro, sin necesidad de basarnos en ninguna librería adicional. Ese desarrollo de Javascript "puro" se conoce generalmente como "***Vanilla Javascript***".

Sin embargo, es habitual que se utilicen **librerías** para un desarrollo web más cómodo y rápido, librerías como **jQuery** o framework como **Angular**, React, Vue,...

> Una **librería** es un conjunto de funciones, o clases y objetos, que nos permite realizar un abanico de tareas habituales para el desarrollo de ciertas necesidades de aplicaciones. Un **framework** se distingue principalmente de una librería porque, además de proveer código para resolver problemas comunes, ofrece una arquitectura que los desarrolladores deben seguir para producir las aplicaciones y asegurarse una mejor calidad del código y mayor facilidad de mantenimiento. Dicho de otro modo, el framework además de ofrecer utilidades diversas, marca un estilo y flujo de trabajo a la hora de desarrollar aplicaciones.

JQuery es la librería más conocida de Javascript. Con jQuery se puede escribir código Javascript que es capaz de ejecutarse sin errores en cualquier navegador, incluso los antiguos, e implementa muchas funcionalidades que se requieren con frecuencia en cualquier sitio web. JQuery permite, además, programar nuevas funcionalidades por medio de *plugins* para hacer cosas tan variadas como validación de formularios, sistemas de plantillas, pases de diapositivas, interfaces de usuario avanzadas y un largo etc. Pero en la actualidad, debido al uso cada vez más extendido de frameworks (que disponen de sus propias librerías), la utilización de la librería JQuery está cayendo en desuso.

Javascript y HTML5

La revolución de Javascript ha llegado con la incorporación de HTML5. A pesar de su nombre, HTML5 incluye varios estándares aparte de ser una revisión del propio lenguaje HTML. De hecho más del 60% de lo que se conoce como HTML5 en realidad son APIs Javascript. Un API es un conjunto de funciones que sirven para llevar a cabo una o muchas tareas. HTML5 incluye diversas API para trabajar, ya no solo con el navegador, sino también con los periféricos o los elementos del dispositivo, como cámara, pantalla, espacio de almacenamiento, GPS, etc.

HTML5 contribuye a estandarizar más Javascript; crea una serie de especificaciones que siguen todos los fabricantes de navegadores para ordenadores y dispositivos y que aseguran que Javascript es igual en todos ellos. HTML5 sirve, por tanto, para ordenadores de escritorio, pero también para todo tipo de dispositivos que nos encontramos actualmente, desde móviles a tablets, smartTV, etc.

Las API del HTML5 nos permiten extender todavía más las posibilidades de Javascript, llegando a situarlo en condiciones similares a las de otros lenguajes de programación. Llegado a este punto es inevitable hablar del concepto de "**Webapp**", que son aplicaciones para móviles y tablets que están basadas en HTML5 (HTML + CSS + Javascript) y que pueden controlar el dispositivo, por medio de las API, de igual modo que los lenguajes de programación nativos.

El problema de HTML5 es que los navegadores antiguos no implementan todas las características del estándar; existen diversas técnicas para aplicar compatibilidad a estos navegadores. En la mayoría de los casos los desarrolladores se decantan por lo que se denomina "*graceful degradation*", que consiste en aplicar técnicas que permitan una "degradación amigable" de las aplicaciones web, de modo que se minimicen los efectos indeseables por la falta de compatibilidad. Estas técnicas incluyen la carga de librerías adicionales para suplir las carencias que tienen los navegadores antiguos o la ejecución de estilos o scripts alternativos. Por ejemplo, **Modernizr** es una librería de Javascript permite la detección de características del navegador y la carga condicional de estilos CSS o de scripts en función de lo compatible, o no, que sea un navegador. Con Modernizr se puede hacer que clientes web obsoletos entiendan perfectamente las nuevas etiquetas del HTML5, que se les apliquen CSS diferentes dependiendo de los navegadores que ven la página y cargar condicionalmente librerías denominadas "***polyfills***" que sirven para implementar de manera no nativa características del HTML5 que son nativas en los navegadores modernos.

Ejemplos de frameworks para el desarrollo de Webapp: **Apache Cordova**, **Phone Gap** o **Ionic**. La ventaja que presentan es que pueden producir apps para Android e iOS con la misma base de código, así como para otros sistemas minoritarios.

Por último, no podemos dejar de hablar de Javascript sin mencionar **NodeJS**. Se trata de un lenguaje de propósito general pero que tiene como particularidad usar el motor de Javascript V8 (el motor de Javascript implementado en el Google Chrome) para la ejecución de los programas. Es capaz de servir para el desarrollo web (en el lado del servidor), y para otro tipo de tareas.

NodeJS es un lenguaje que tiene unas características diferentes a otros lenguajes; como su característica más peculiar encontramos lo que se llama programación "asíncrona" que no es más que la capacidad de realizar acciones que lleven un tiempo para la ejecución sin necesidad de mantener procesos en estado de espera. Esto provoca que su ejecución sea bastante ligera, aunque también complica algo las cosas para las personas que están acostumbradas a que los

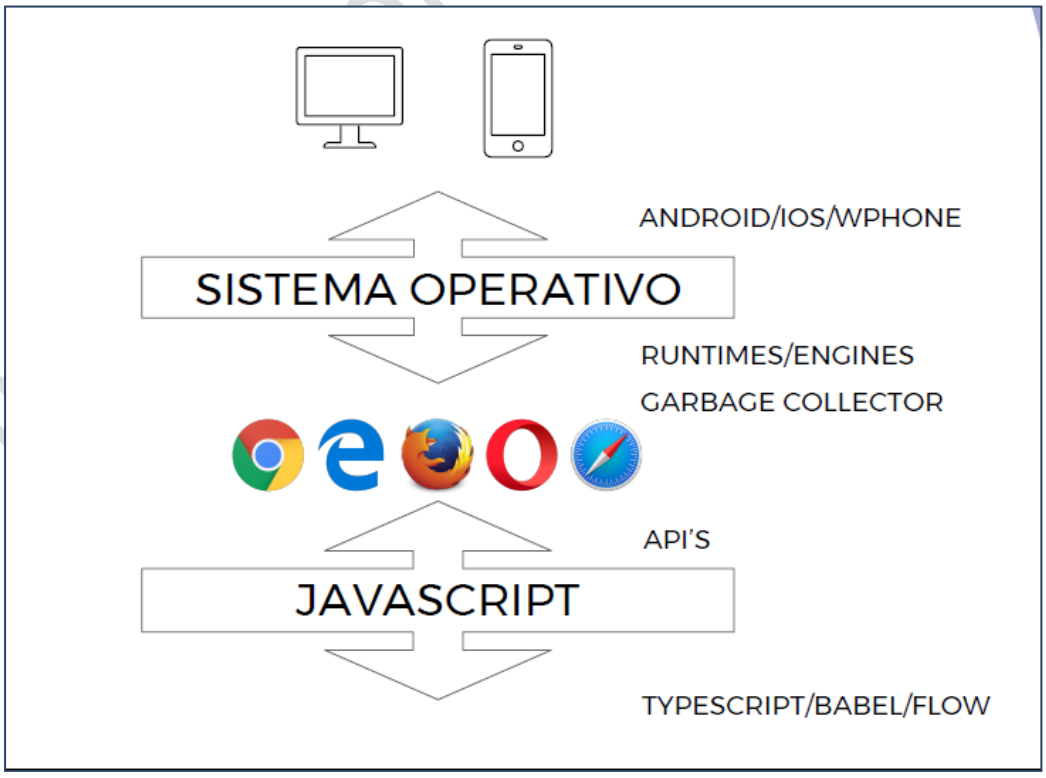
programas siempre se ejecutan siguiendo un flujo fijo de instrucciones, tal como están escritos. Muchas herramientas del día a día de los desarrolladores están programadas con NodeJS, como los gestores de paquetes "*npm*" o "*bower*". También muchas herramientas para desarrollo *frontend* están programadas en NodeJS. Además existen diversos frameworks como SailsJS o ExpressJS que se pueden usar para aplicar NodeJS en el desarrollo web.

5. Ejecución en Javascript

Sabemos que cualquier programa (independientemente del lenguaje en que esté escrito) se ejecuta, en última instancia, en el hardware del equipo: ordenador, tablet, teléfono inteligente, ...

Por encima del hardware se localiza el sistema operativo (SO del ordenador de escritorio/portátil o SO del dispositivo móvil).

En el caso de Javascript, la siguiente capa dónde se ejecuta un programa es el **navegador**. Un navegador es una aplicación muy compleja que debe, entre otras tareas, ejecutar scripts y para ello debe gestionar dos actividades importantes tales como el **runtime/engine** (motor de ejecución de Javascript) y el **garbage collector** (gestor automático de memoria)

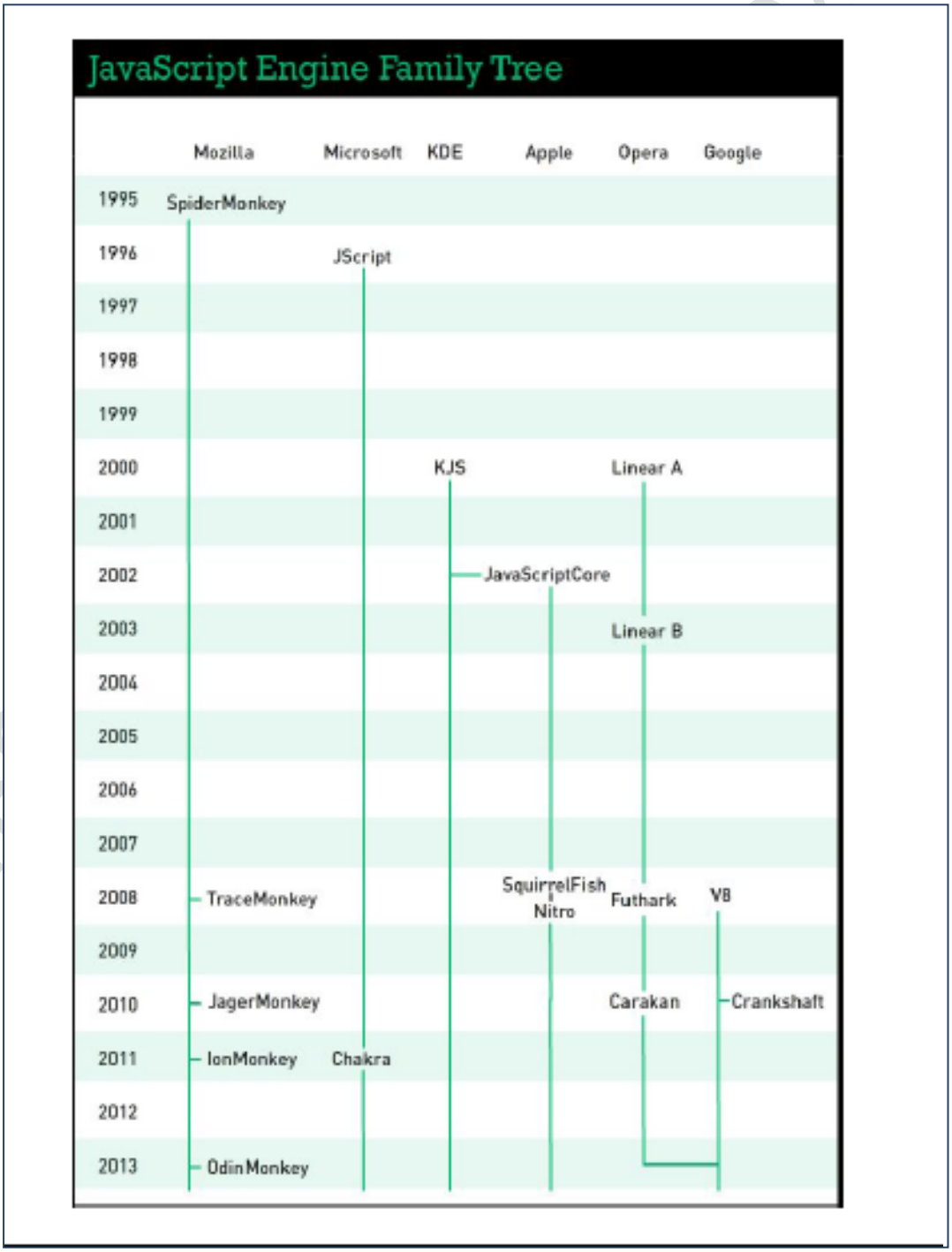


El hardware, sistema operativo y versión del navegador existentes en el mercado son muy diferentes en cuanto a potencia y rendimiento. Si queremos que nuestras aplicaciones sean lo más "universales" posibles, no deberíamos perder de vista esta realidad.

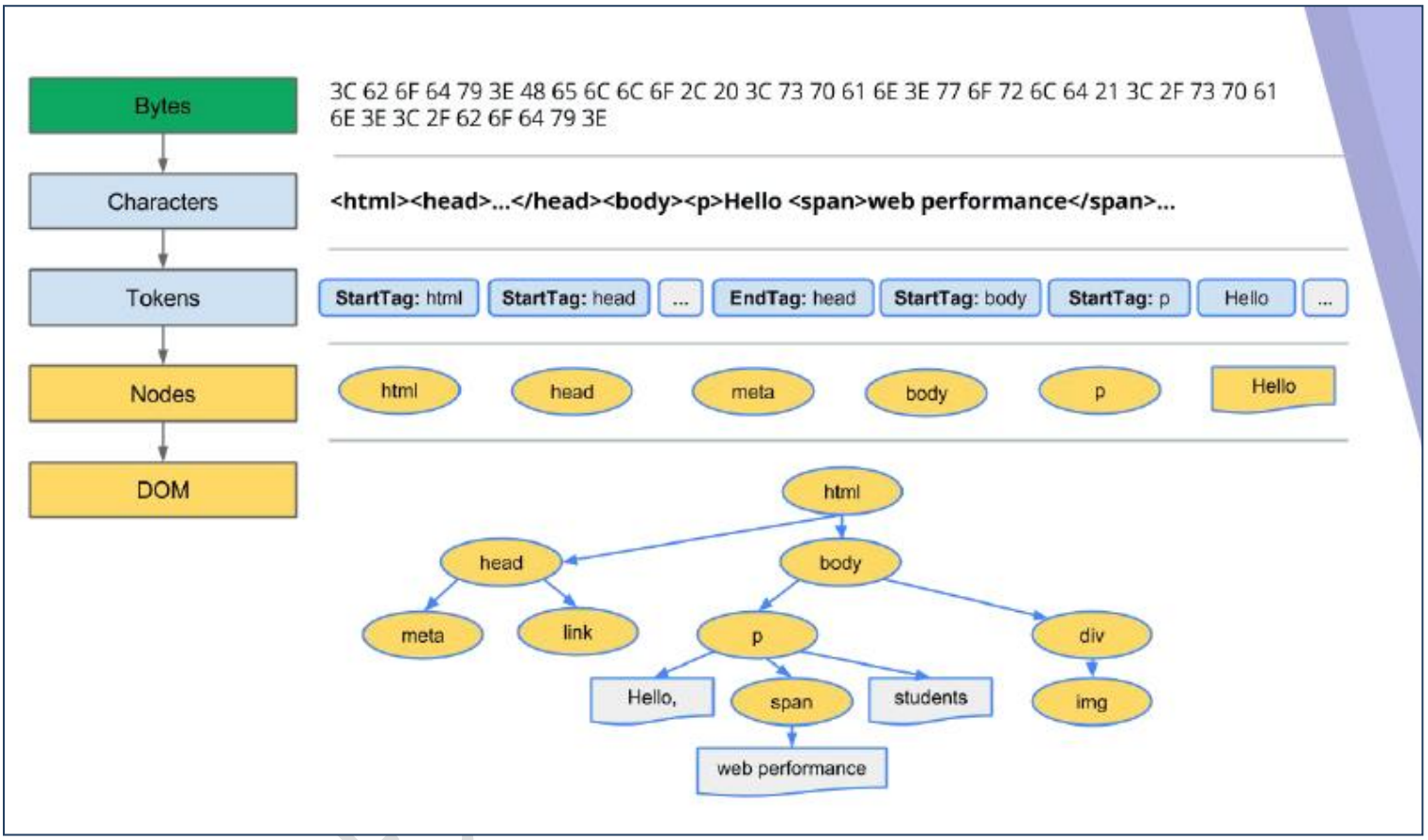
Runtime/engine

Es la parte del navegador que ejecuta código javascript.

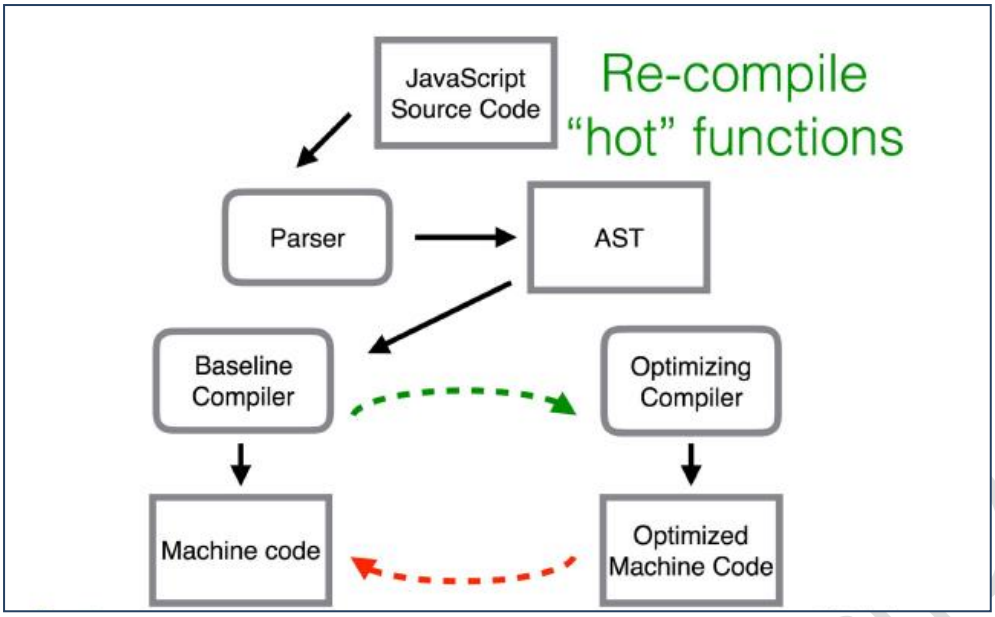
Evolución de runtimes en distintos navegadores:



El navegador, cuando recibe una página HTML, lo hace en forma de bytes, que transforma en caracteres (las etiquetas html). A continuación, los caracteres se parsean, identificando las etiquetas (*tokens*), verificando que la estructura recibida es correcta y creando los **nodos** correspondientes a dichas etiquetas que van a constituir el árbol **DOM** (modelo de objetos del documento). Los nodos que componen el DOM son objetos que encapsulan las etiquetas HTML; estos nodos podrán ser manejados desde el código Javascript, proporcionando dinamismo a la página web en el lado del cliente.



Cuando el navegador identifica que el código recibido dentro del página web es Javascript, lo descarga y realiza un proceso parecido al anterior: el código fuente se parsea y es transformado en un árbol de sintaxis abstracto (**AST**). AST es un lenguaje intermedio que necesita el *runtime* para identificar las tareas programadas y poder ejecutarlas. Después del parseo, el árbol AST se interpreta y optimiza y es ejecutado por el equipo.



Con la herramienta **AST explorer** podemos visualizar el proceso de creación del árbol AST:

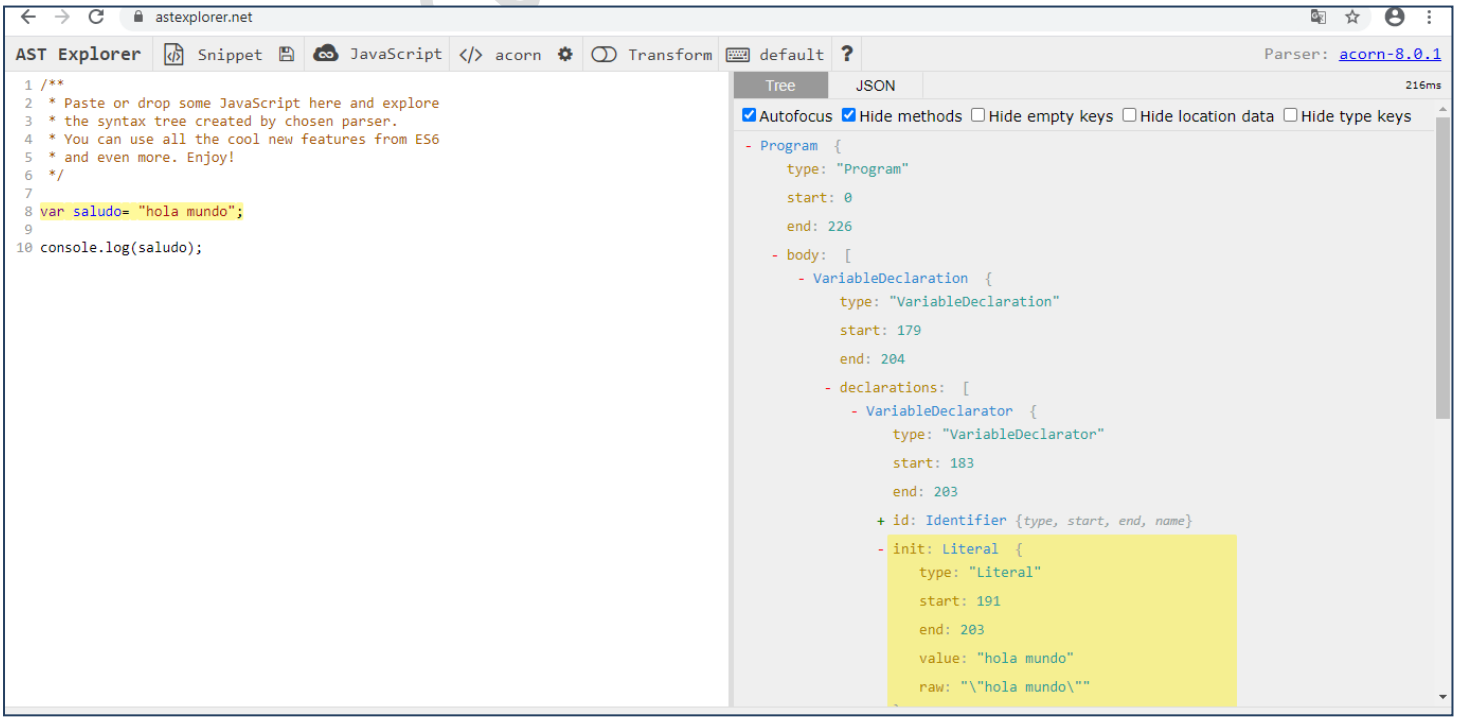
https://astexplorer.net/

En el siguiente ejemplo podemos ver el árbol AST generado para el código fuente Javascript que se indica a continuación:

```
var saludo= "hola mundo";

console.log(saludo);
```

En este ejemplo se declara la variable *saludo*, de tipo *string*, con valor "*hola mundo*" y, a continuación, se muestra el valor de la variable por la consola del navegador.



En ventana de la izquierda vemos el código Javascript y en la ventana derecha el árbol AST generado, capaz de ser interpretado por el navegador web.

Garbage collector

Es el gestor de memoria del navegador; realiza limpiezas automáticas para evitar agotarla. Si la memoria colapsa, finalizará el proceso en ejecución.

El recolector considera "basura" aquellos elementos (objetos y datos primitivos) que quedan sin referencia en la memoria.

6. Referencias bibliográficas

- https://programacionwebisc.wordpress.com/2-1-arquitectura-de-las-aplicaciones-web/
- https://sites.google.com/site/jojooa/informatica-tecnologia/definicion-de-los-lenguajes-de-programacion-del-lado-del-cliente-que-son-los-lenguajes-de-programacion-del-lado-del-cliente
- https://desarrolloweb.com/manuales/27/
- https://desarrolloweb.com/javascript/
- https://openwebinars.net/