

# 01. Introducción

Desarrollo web en entorno cliente

José María Torresano  
Septiembre 2020  
daw.cierva@gmail.com

**HTML**



El lenguaje

**CSS**



El diseño

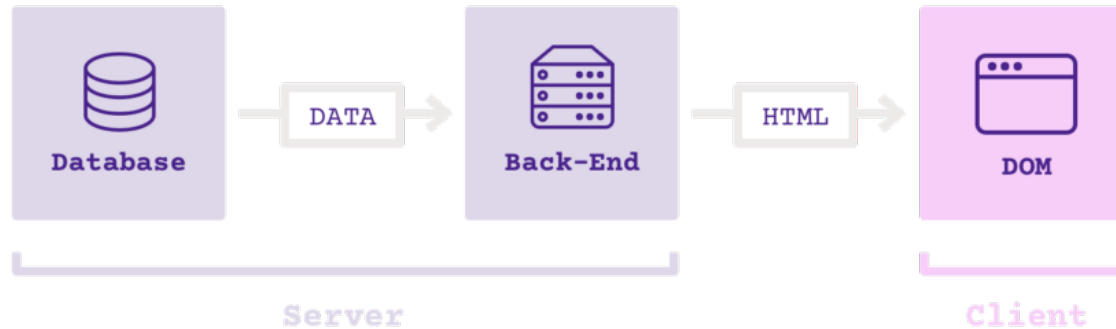
**JAVASCRIPT**



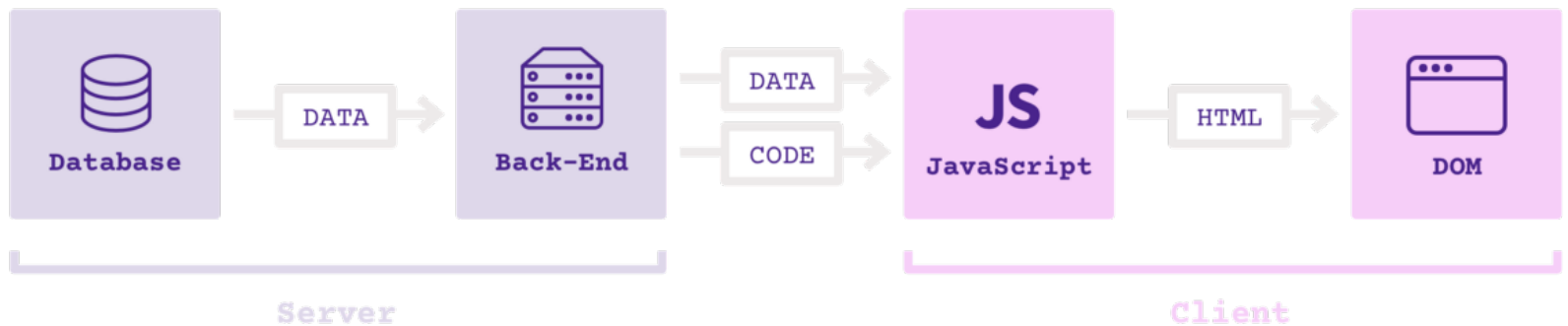
La interacción

## Tipos de desarrollos en el lado cliente

Clásico



SPA



SPA – con estado



# Espectro

Desarrolladores

Diseñadores



Sistemas/SS.OO.

Backend (Servidor)

Móviles

Frontend (Cliente)

JS Ocasional

CSS/HTML Ocasional

Diseñador UX/UI

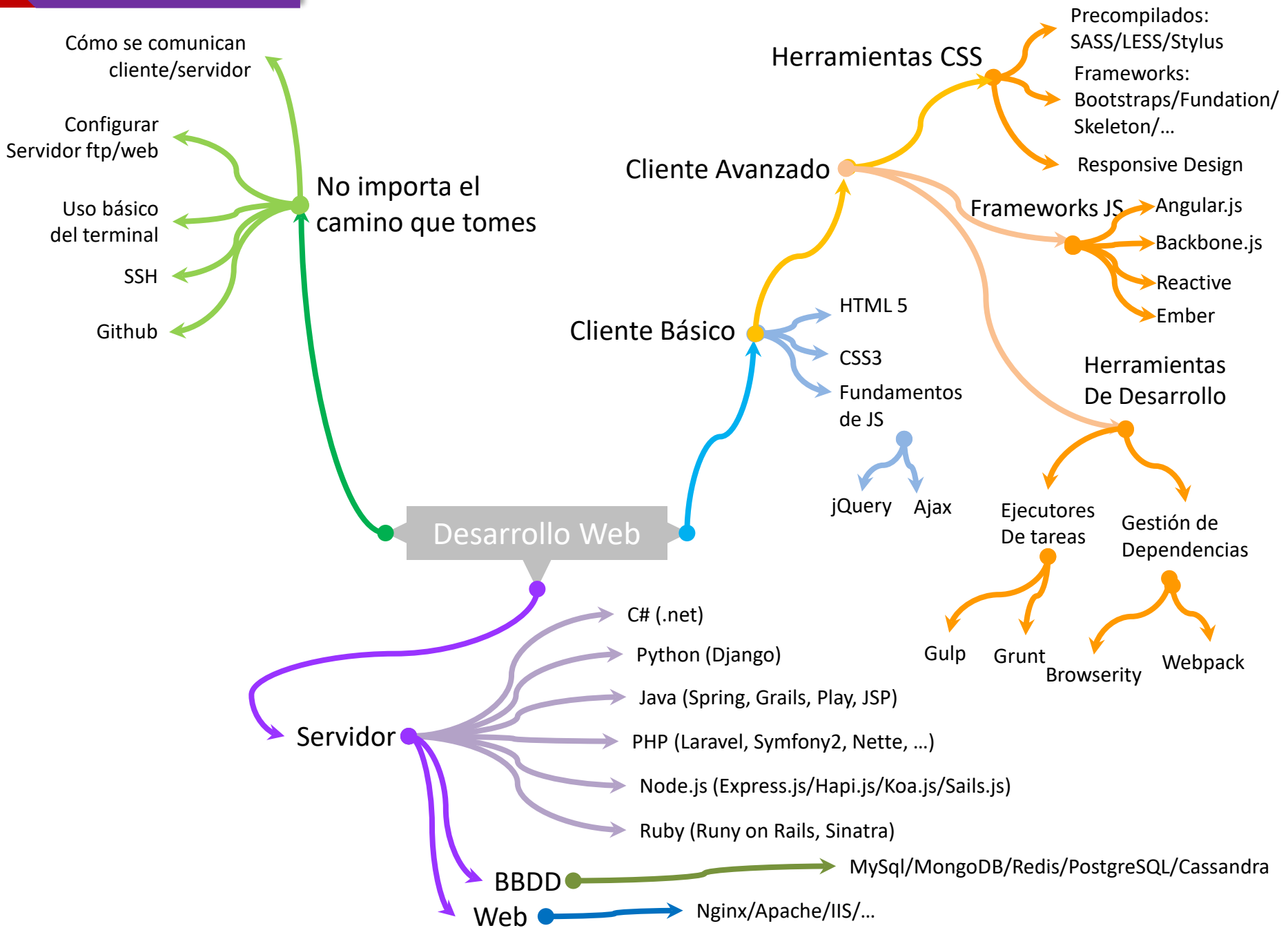
Diseñador Gráfico

Fotógrafo

## Intersección







**WEB DESIGN / WEB DEVELOPMENT**



# NAVEGADORES

# Uso del navegador







Navegador	Agosto 2018	Agosto 2019	Agosto 2020
	60%	64%	66%
	14%	15%	17%
	5%	4%	4%
	3%	2%	2%

En el **mundo**, **todos** los dispositivos



# Uso del navegador






Navegador	Agosto 2018	Agosto 2019	Agosto 2020
	68%	71%	70%
	11%	9,5%	8,2%
	7%	5,8%	2,6%
	4%	6%	8,3%

En el **mundo**, en **escritorio**

# Uso del navegador






Navegador	Agosto 2018	Agosto 2019	Agosto 2020
	61%	65%	47%
	25%	23%	40%
	12%	12%	11,4%

En el **mundo**, en **tableta**

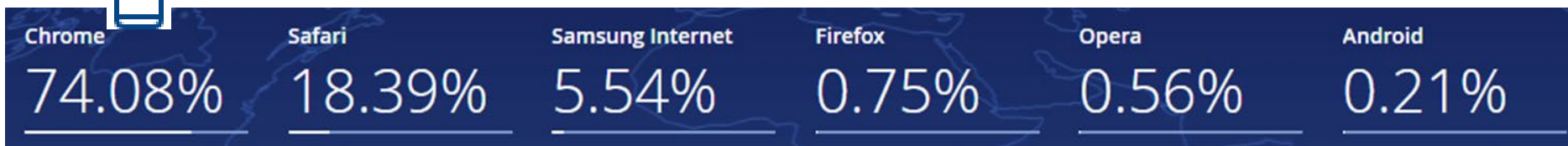
# Uso del navegador



Navegador	Agosto 2018	Agosto 2019	Agosto 2020
	56%	61%	64%
	19%	20%	23%
	11%	6%	2,5%

En el **mundo**, en **móviles**





# Uso del navegador



**En España, actual.**

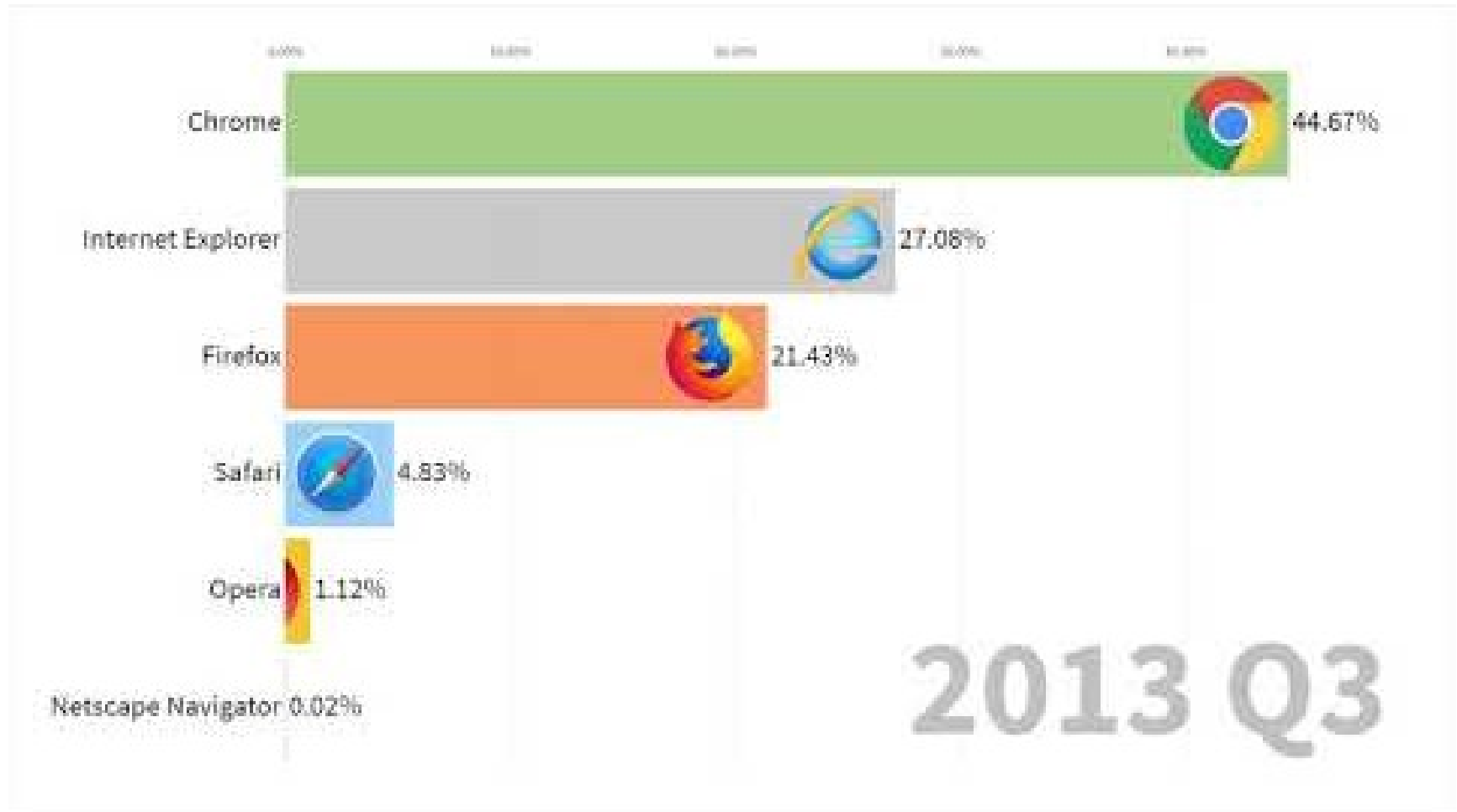
# Uso del navegador



Navegador	Agosto 2018	Agosto 2019	Agosto 2020
	67%	71%	69,43%
	13%	12%	11%
	7%	7%	2,4%
	6%	4%	8%

En el **España**, en **escritorio**

# Uso del navegador



<https://www.youtube.com/watch?v=es9DNe0l0Qo&feature=youtu.be>

# HERRAMIENTAS

# Herramientas de desarrollo





# Últimas versiones (para desarrolladores)



Chrome Canary



Firefox Developer



Webkit Nightly



# Herramientas de desarrollo

**Chrome Developer Tools** te permite profundizar en las entrañas de una página web y revisar casi todo, desde su estructura hasta los recursos utilizados. En general puedes:

- Obtener una visión general de los **estilos** utilizados en una página y que estilo se aplica a cada elemento.
- Visualizar y modificar el **código** correspondiente a los elementos HTML
- Modificar un estilo y ver los **cambios** en el navegador en tiempo real
- **Ejecutar** código JavaScript en el contexto de la página
- Ver las **peticiones** HTTP realizadas por el navegador
- Identificar cuellos de botella que afecten al **rendimiento**
- Consultar **métricas** de rendimiento
- Investigar los **recursos** offline para averiguar que datos de la página se almacenan localmente

<https://developers.google.com/web/tools/chrome-devtools/>



# Herramientas de desarrollo. Controles de Interfaz

Inspeccionar un elemento

Diferentes paneles de herramientas

Errores y advertencias de JS

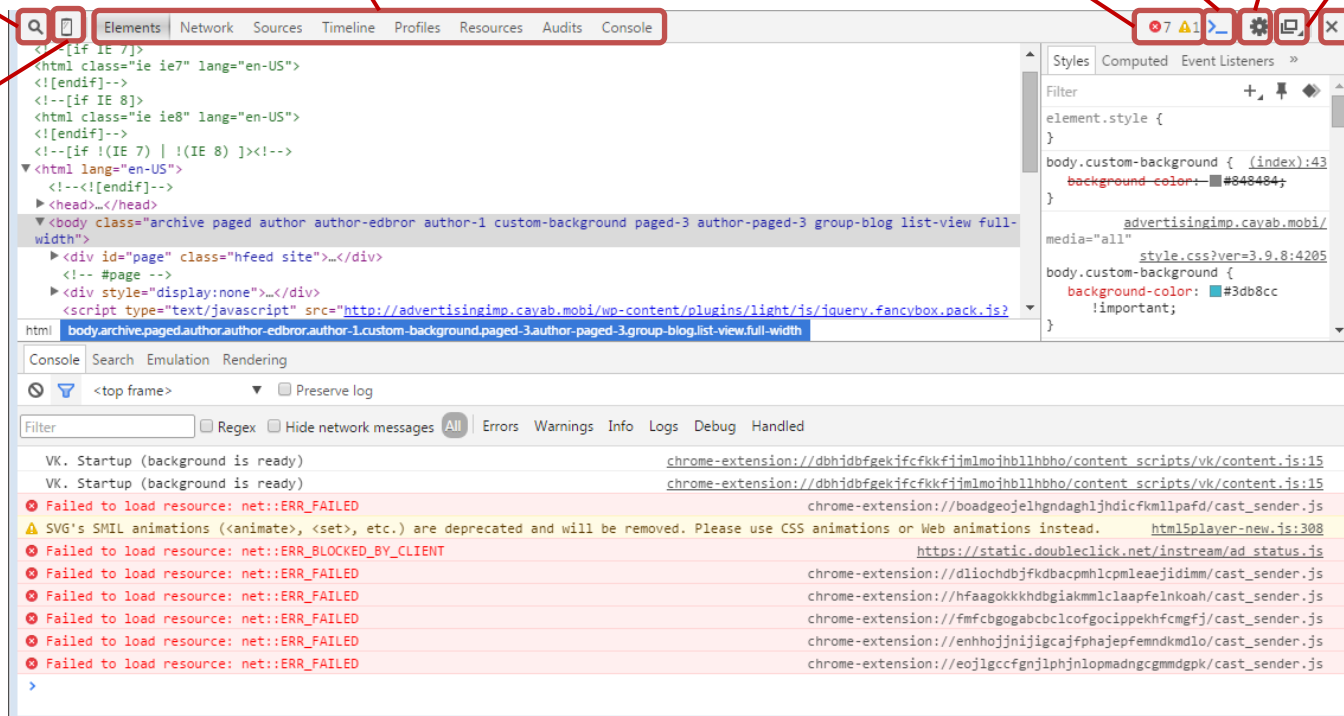
Mostrar/Ocultar consola

Configuración

Anclar/desanclar

Elección de dispositivo

Cerrar



# FUNCIONAMIENTO

## Función principal de un navegador

La función principal de un navegador es solicitar al servidor los recursos web que elija el usuario y mostrarlos en una ventana.

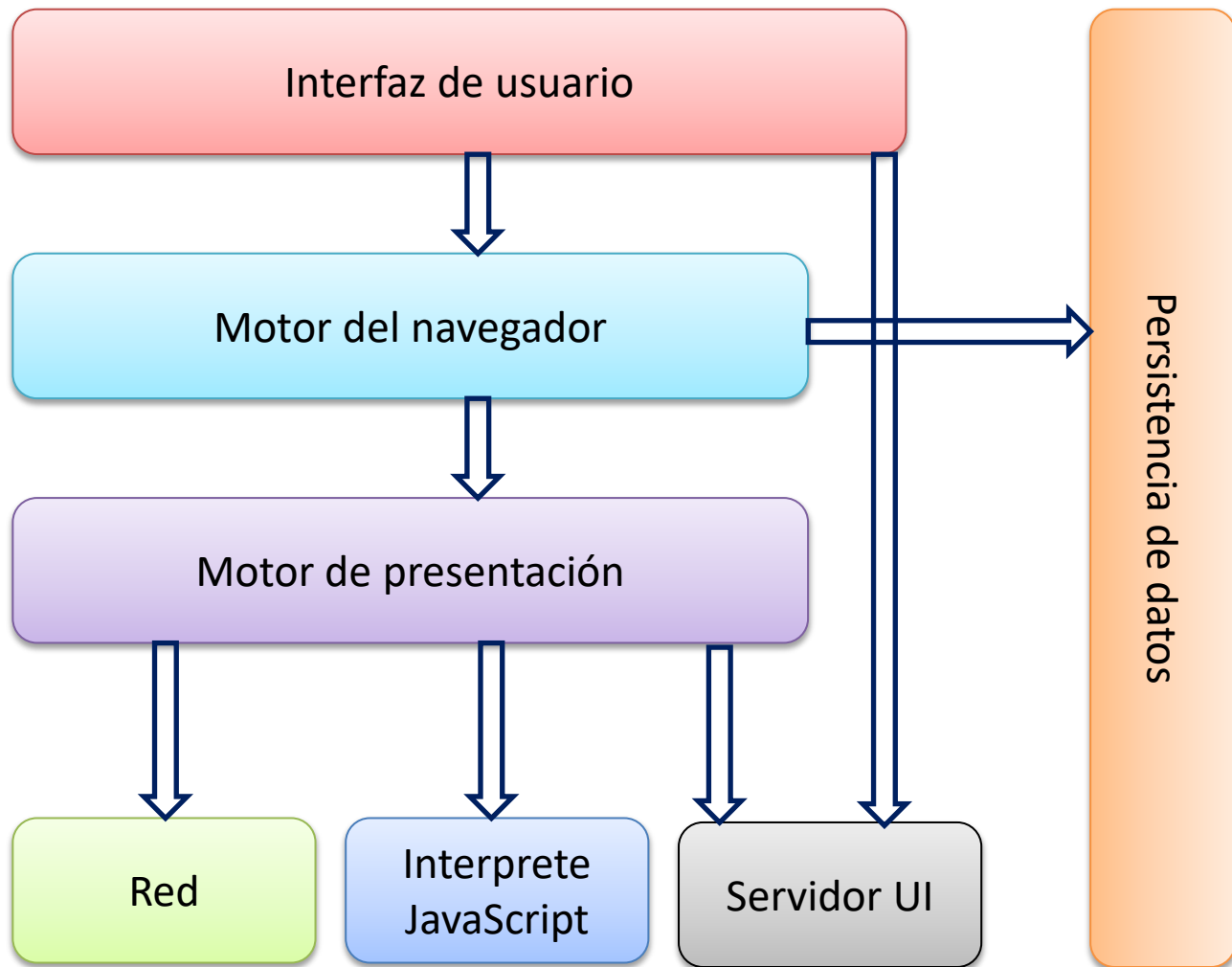
El recurso suele ser un documento HTML, pero también puede ser un archivo PDF, una imagen o un objeto de otro tipo. El usuario especifica la ubicación del recurso mediante el uso de una URI (siglas de Uniform Resource Identifier, identificador uniforme de recurso).



La forma en la que el navegador interpreta y muestra los archivos HTML se determina en las especificaciones de CSS y HTML. Estas especificaciones las establece el consorcio **W3C** (World Wide Web Consortium), que es la organización de estándares de Internet



Durante años, los navegadores cumplían solo una parte de las especificaciones y desarrollaban sus propias extensiones. Esto provocó graves problemas de compatibilidad para los creadores de contenido web. En la actualidad, la mayoría de los navegadores cumplen las especificaciones en mayor o menor grado.



**Interfaz de usuario**

Incluye la barra de direcciones, el botón de avance/retroceso, el menú de marcadores, etc. (en general, todas las partes visibles del navegador, excepto la ventana principal donde se muestra la página solicitada).

**Motor del navegador**

Coordina las acciones entre la interfaz y el motor de renderización (presentación).

**Motor de presentación**

Muestra el contenido solicitado. Por ejemplo, si el contenido solicitado es HTML, será el responsable de analizar el código HTML y CSS y de mostrar el contenido analizado en la pantalla.

**Red**

Es responsable de las llamadas de red, como las solicitudes HTTP. Tiene una interfaz independiente de la plataforma y realiza implementaciones en segundo plano para cada plataforma.

**Servidor UI**

Representa widgets básicos, como ventanas y cuadros combinados. Muestra una interfaz genérica que no es específica de ninguna plataforma. Utiliza métodos de la interfaz de usuario del sistema operativo en segundo plano.

**Interprete JavaScript**

Permite analizar y ejecutar el código JavaScript.

**Persistencia de datos**

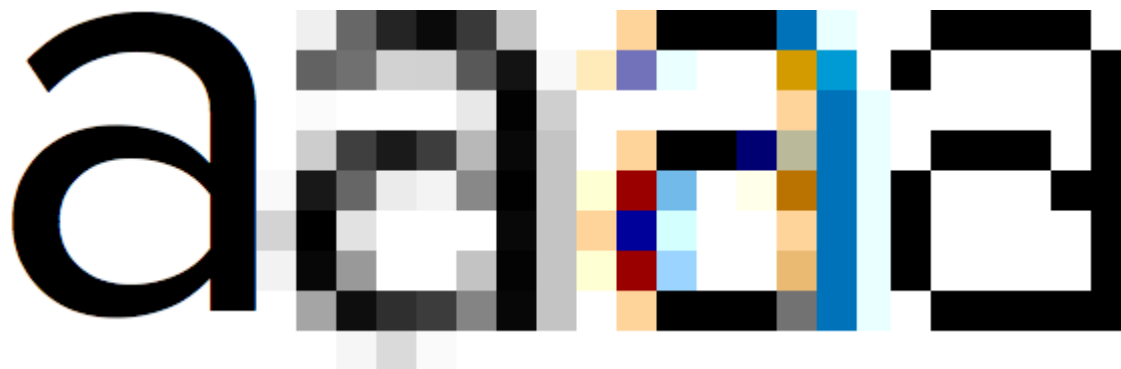
El navegador necesita guardar todo tipo de datos en el disco duro (cookies, ...). La especificación HTML5 define el concepto de "base de datos web", que consiste en una completa (aunque ligera) base de datos del navegador.



## Motor de presentación (renderización)

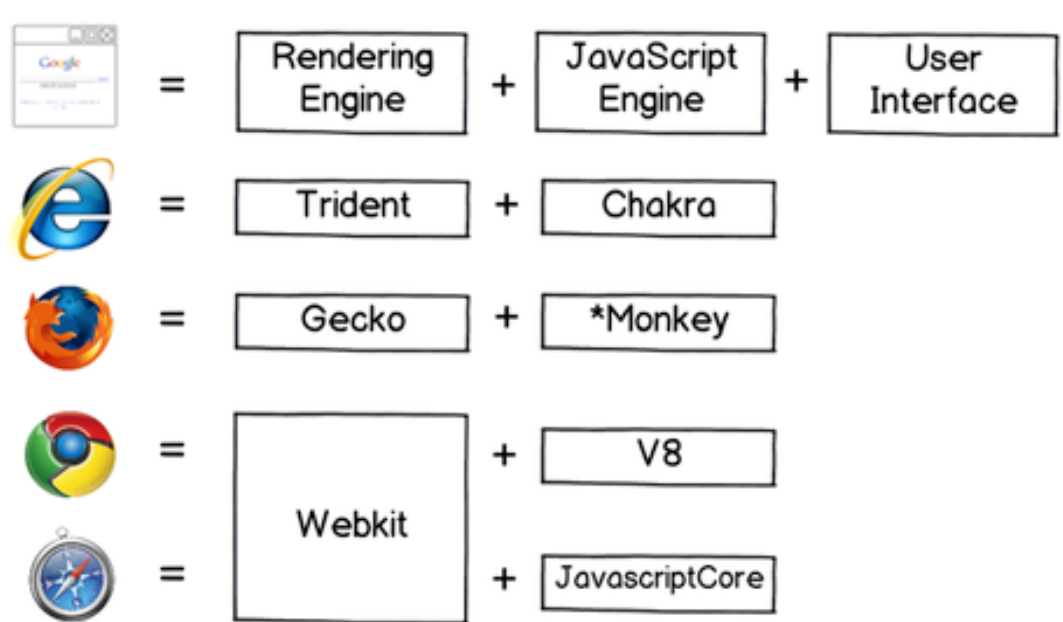
La responsabilidad del motor de renderización es "renderizar", es decir, mostrar el contenido solicitado en la pantalla del navegador.

De forma predeterminada, el motor de *renderización* puede mostrar imágenes y documentos HTML y XML. Puede mostrar otros tipos mediante el uso de complementos (o extensiones); por ejemplo, puede mostrar documentos PDF mediante un complemento capaz de leer archivos PDF. Sin embargo, en este capítulo nos centraremos en su uso principal: mostrar imágenes y código HTML con formato definido con CSS.



## Motor de presentación (renderización)

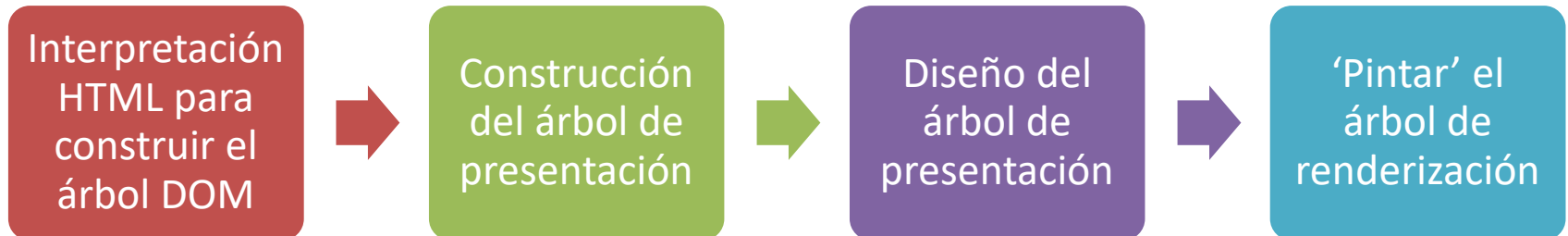
Los navegadores de referencia (Firefox, Chrome y Safari) están basados en dos motores de renderización. Firefox utiliza Gecko, un motor propio de Mozilla. Tanto Safari como Chrome utilizan WebKit. IEExplorer utiliza Trident; Spartan HTMLEdge.



## El flujo principal

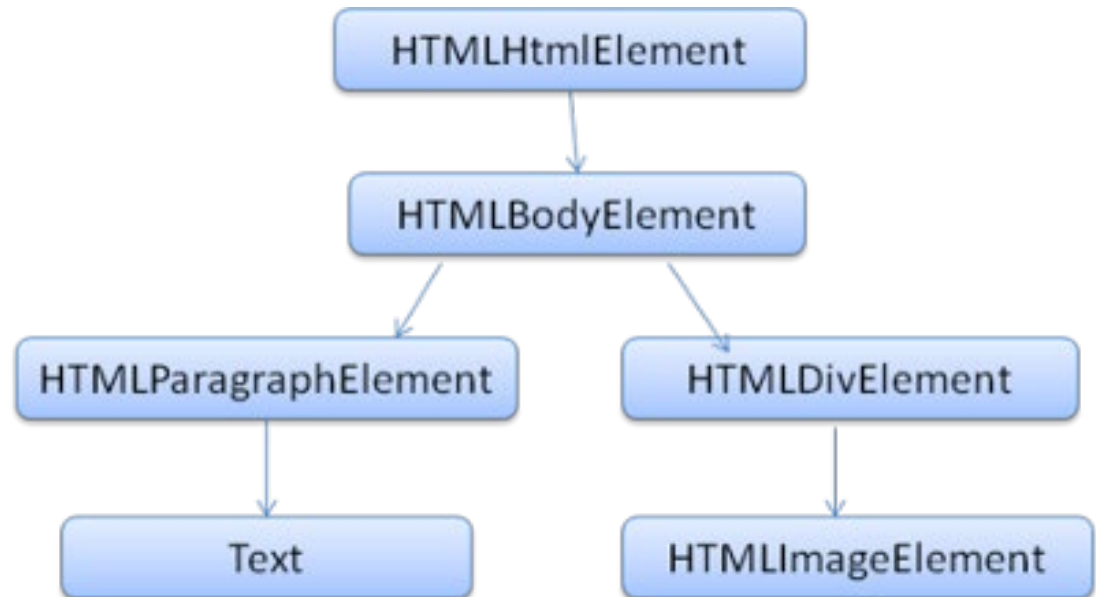
El motor de renderización empieza recibiendo el contenido del documento solicitado desde la capa de red, normalmente en fragmentos de 8.000 bytes.

A continuación, el motor de renderización realiza este flujo básico

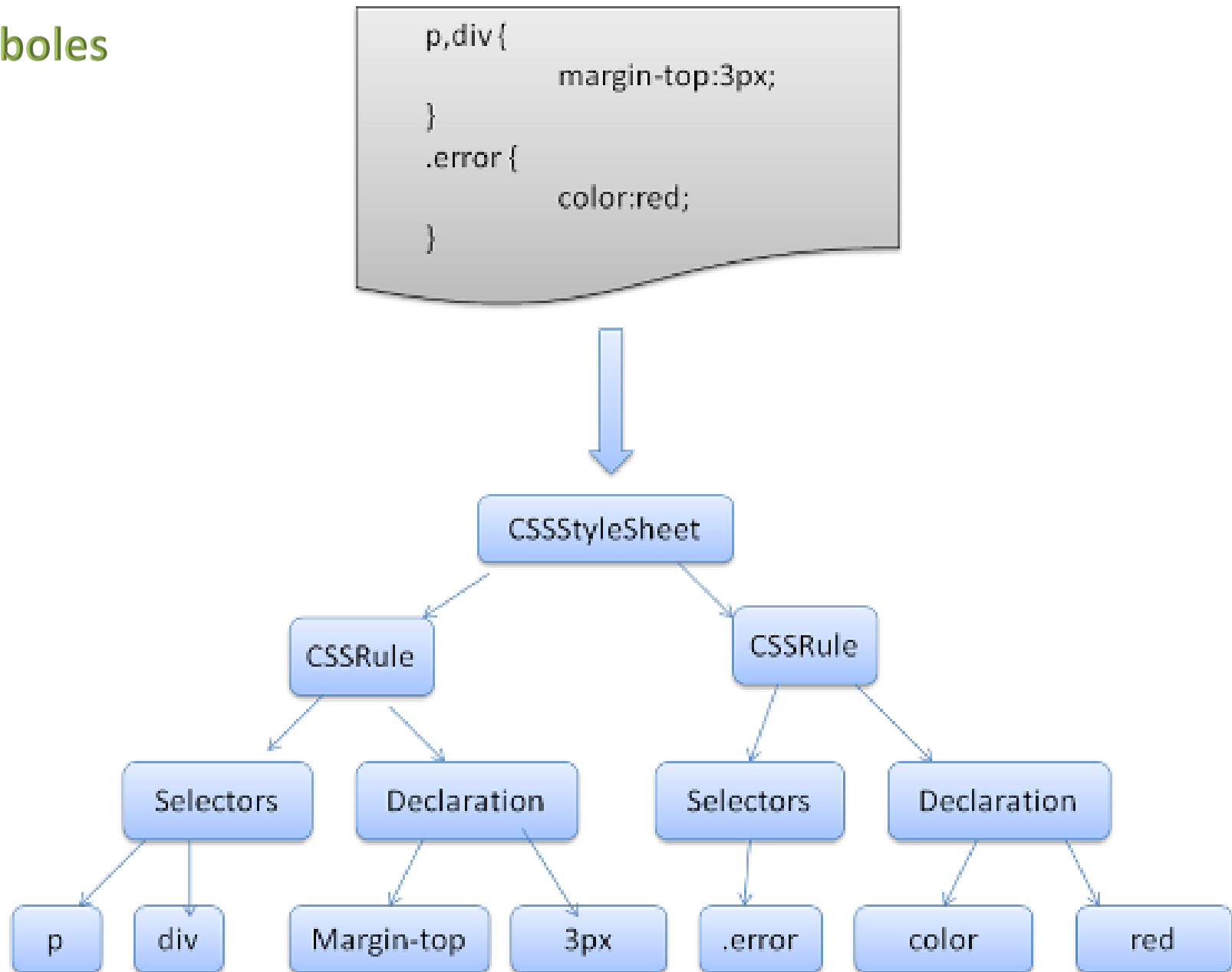


# Árboles

```
<html>
  <body>
    <p>
      ¡Hola Mundo!
    </p>
    <div> <img src "ejemplo.png"/></div>
  </body>
</html>
```



## Árboles



### Interpretación HTML para construir el árbol DOM

El motor de renderización empieza a analizar el documento HTML y convierte las etiquetas en nodos DOM en un árbol denominado "árbol de contenido". Analiza los datos de estilo, tanto en los archivos CSS externos como en los elementos de estilo. Los datos de estilo, junto con las instrucciones visuales del código HTML, se utilizan para crear otro árbol: el árbol de renderización.

### Construcción del árbol de presentación

El árbol de renderización contiene rectángulos con atributos visuales, como el color y las dimensiones. Los rectángulos están organizados en el orden en el que aparecerán en la pantalla.

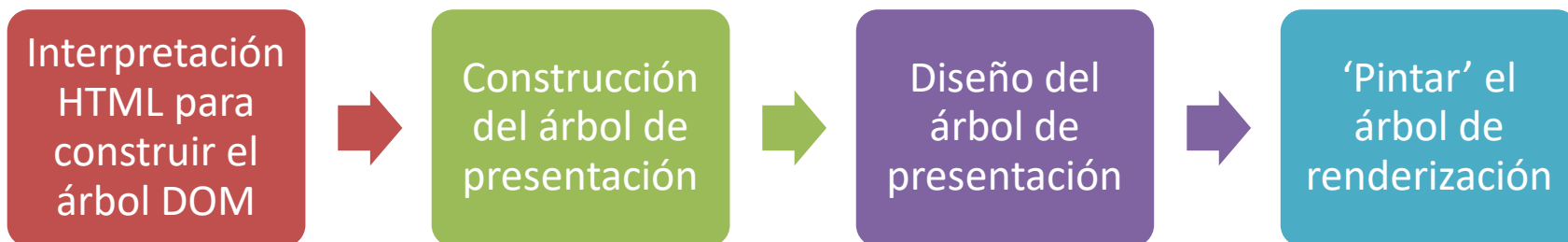
### Diseño del árbol de presentación

Una vez construido el árbol de renderización, se inicia un proceso de "diseño". Esto significa que a cada nodo se le asignan las coordenadas exactas del lugar de la pantalla en el que debe aparecer.

### ‘Pintar’ el árbol de renderización

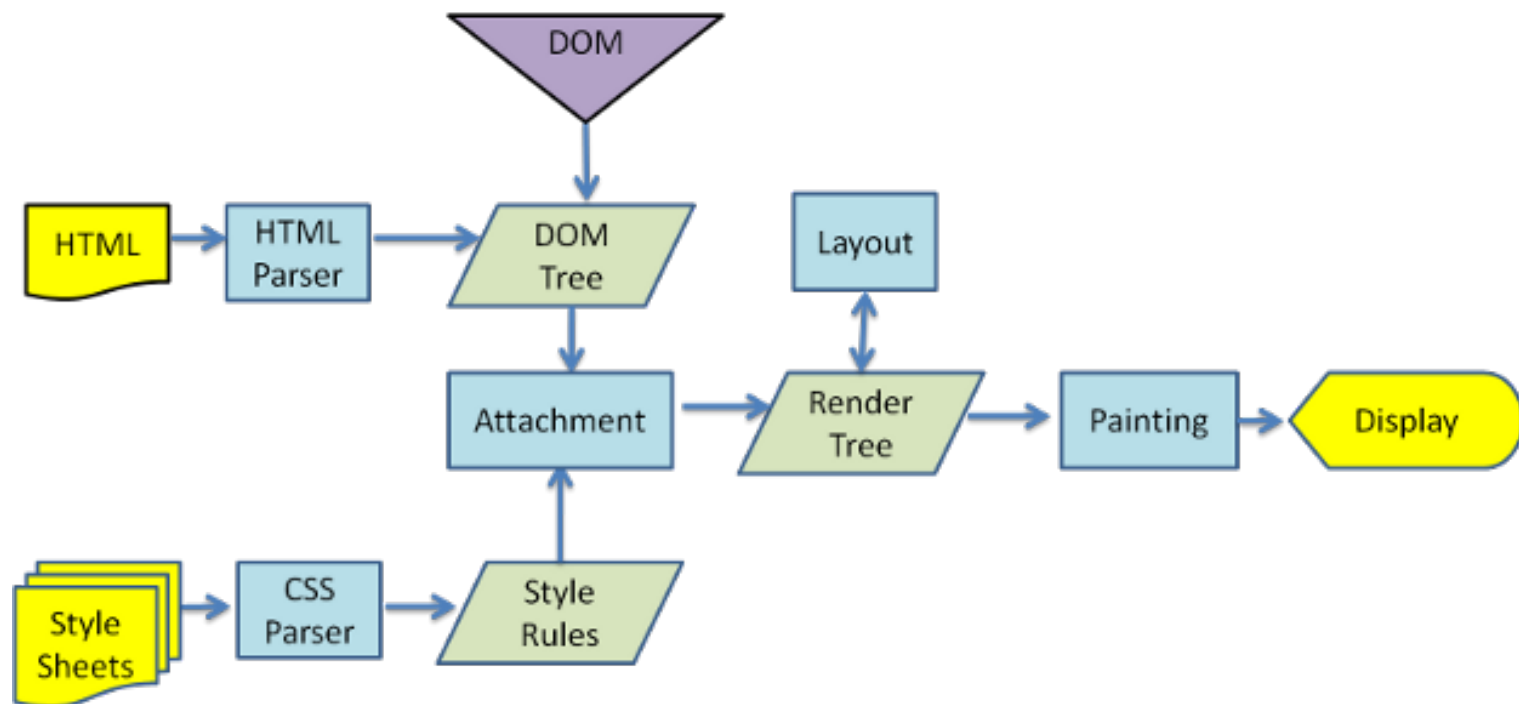
La siguiente fase es la de pintura, en la que se recorre el árbol de renderización y se pinta cada uno de los nodos utilizando la capa de servidor de la interfaz de usuario

## El flujo principal



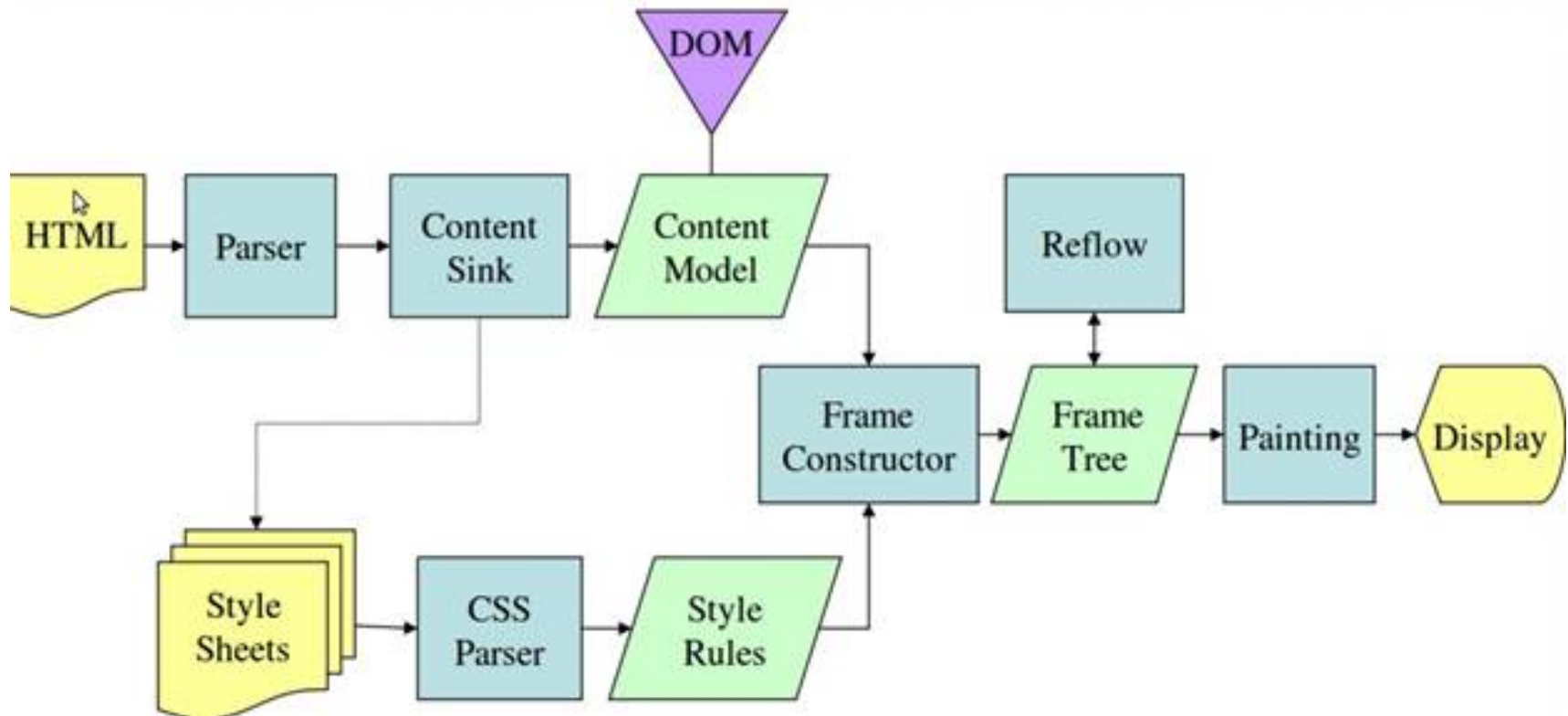
Es importante comprender que se trata de un proceso gradual. Para mejorar la experiencia del usuario, el motor de renderización intentará mostrar el contenido en pantalla lo antes posible. No esperará a que se analice el código HTML para empezar a crear y diseñar el árbol de renderización. Se analizarán y se mostrarán algunas partes del contenido, mientras que se sigue procesando el resto del contenido que llega de la red.

## El flujo principal: WebKit





## El flujo principal: Gecko



# **ORDEN DE PROCESAMIENTO DE SECUENCIAS DE COMANDOS Y HOJAS DE ESTILO**

## Secuencia de comandos

El modelo de la Web es síncrono.

Los autores esperan que las secuencias de comandos se analicen y se ejecuten inmediatamente cuando el analizador llega a la etiqueta `<script>`. El análisis del documento se detiene hasta que la secuencia de comandos se ejecuta. La secuencia de comandos es externa, por lo que antes es necesario recuperar el recurso de la red.

Esta acción se realiza también de una forma síncrona, es decir, que el análisis se detiene hasta que se recupera el recurso. Este modelo se utilizó durante muchos años y está incluido en las especificaciones de HTML 4 y 5.

Los autores pueden marcar la secuencia de comandos como "aplazada". De ese modo, no se detiene el análisis del documento y la secuencia se ejecuta una vez que se ha completado el análisis.

## Análisis especulativo

Tanto WebKit y Firefox utilizan esta optimización. Al ejecutar las secuencias de comandos, otro subproceso analiza el resto del documento, busca en la red otros recursos necesarios y los carga.

De esta forma, los recursos se pueden cargar mediante conexiones paralelas, lo que mejora la velocidad general.

**Nota:** el analizador especulativo no modifica el árbol de DOM (de eso se encarga el analizador principal), solo analiza las referencias a recursos externos, como secuencias de comandos externas, hojas de estilo e imágenes.

## Hojas de estilo

Tienen un modelo diferente.

Conceptualmente parece que, dado que las hojas de estilo no modifican el árbol de DOM, no hay razón para esperarlas y detener el análisis del documento. Sin embargo, se produce una incidencia cuando las secuencias de comandos solicitan información de estilo durante la fase de análisis del documento.

Esta acción se realiza también de una forma síncrona, es decir, que el análisis se detiene hasta que se recupera el recurso. Este modelo se utilizó durante muchos años y está incluido en las especificaciones de HTML 4 y 5.

Firefox bloquea todas las secuencias de comandos si todavía se está cargando y analizando una hoja de estilo. WebKit bloquea las secuencias de comandos solo cuando intentan acceder a determinadas propiedades de estilo que se pueden ver afectadas por las hojas de estilo descargadas.

# Evolución de la web

<http://evolutionofweb.appspot.com/>

