UT 2 - Crear interfaces web en HTML

CFGS Desarrollo de Aplicaciones Web
Diseño de Interfaces Web

Componentes principales

- **Interfaz de usuario:**
- **El motor del navegador**
- **El motor de renderización**
- **Herramientas de redes**
- **Backend de la IU**
- **Intérprete de JavaScript. Almacenamiento de datos.**

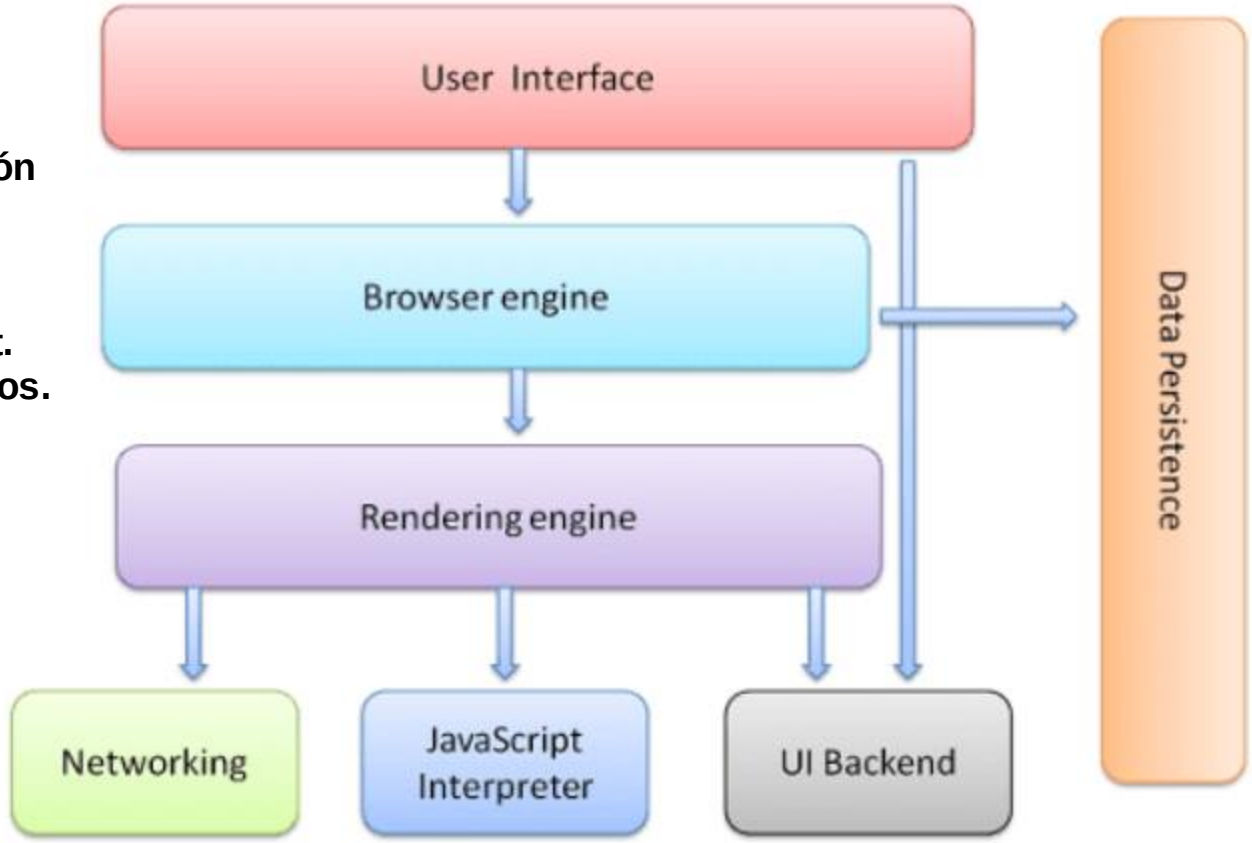


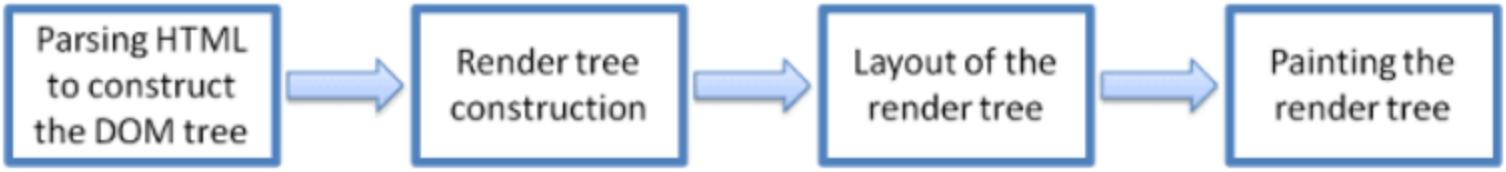
Figura 1: Componentes del navegador

Componentes principales

- **Interfaz de usuario:** Difiere de unos navegadores a otros pero es la herramienta que permite al usuario comunicarse con todas las aplicaciones web. Incluye la barra de direcciones, el botón Atrás/Adelante, el menú de favoritos, etc.

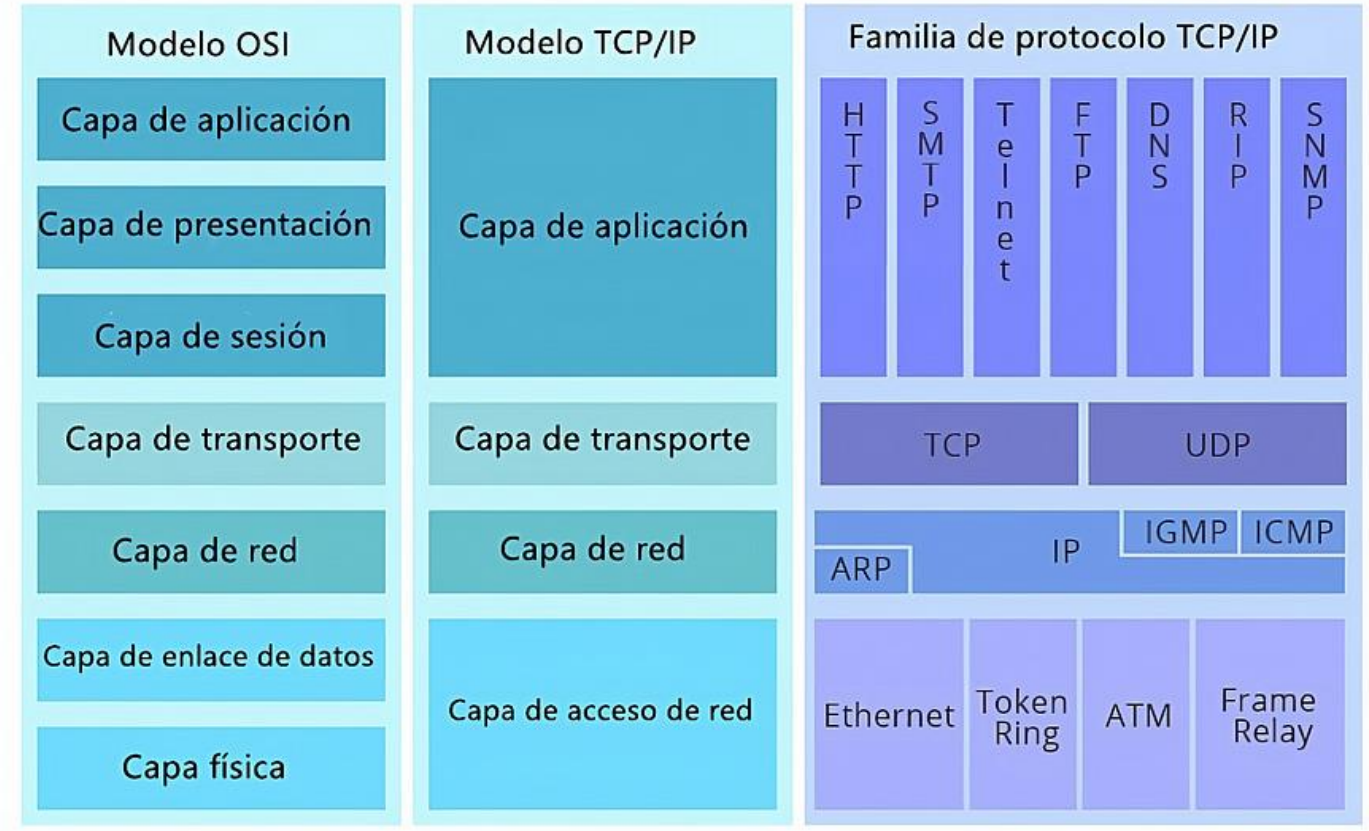
- **El motor del navegador:** Es el proceso que regula las acciones entre entre la IU y el motor de renderización.
- Además, esta parte permite, mostrar imágenes y documentos HTML y XML. Puede mostrar otros tipos de datos a través de complementos o extensiones. por ejemplo, para mostrar documentos PDF con un complemento de lector de PDF.

- **El motor de renderización:** Renderizar, es mostrar el contenido solicitado en la pantalla del navegador.
- Los diferentes navegadores utilizan distintos motores de representación: Internet Explorer usa Trident, Firefox usa Gecko y Safari usa WebKit. Chrome y Opera (de la versión 15) utilizan Blink



El motor de renderizado

El motor de renderización comenzará a obtener el contenido del documento solicitado de la capa de red. Por lo general, esto se hace en fragmentos de 8 KB.



El motor de renderizado

Estructura:

Comenzará a analizar el documento HTML y convertirá los elementos en nodos **DOM (Document Object Model)** en un árbol denominado "árbol de contenido".

El motor analizará los datos de estilo, tanto en archivos CSS externos como en elementos de estilo. La información de estilo junto con las instrucciones visuales en el código HTML se usará para crear otro árbol: **el árbol de renderización o representación**.

Relación del árbol de representación con el árbol del DOM

El procesamiento de las etiquetas html y body da como resultado la construcción de la raíz del árbol de renderización.

Para compilar el árbol de renderización, es necesario calcular las propiedades visuales de cada objeto de renderización. Para ello, se calculan las propiedades de estilo de cada elemento.

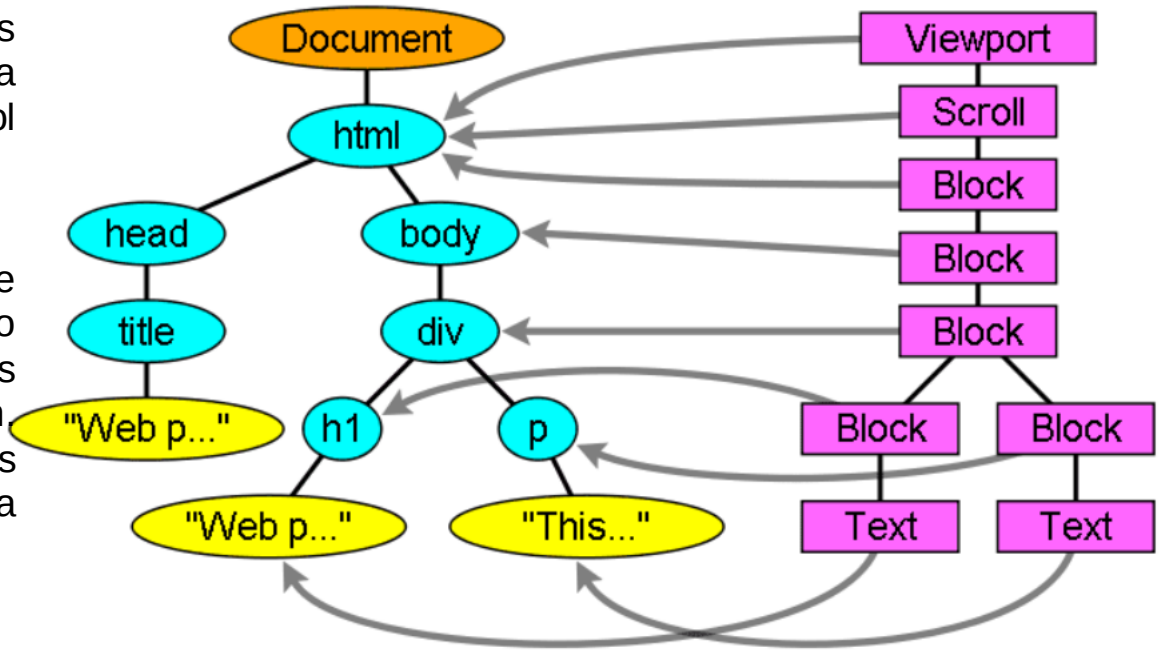


Figura 13: Árbol de representación y árbol del DOM correspondiente. La "Ventana gráfica" es el bloque contenedor inicial. En WebKit, será la "RenderView". objeto

El motor de renderizado

Cálculo de estilo:

Los datos de estilo son una construcción muy grande que contiene las numerosas propiedades de estilo, lo que puede causar problemas de memoria.

Encontrar las reglas de coincidencia para cada elemento puede causar problemas de rendimiento si no está optimizado.

```
div div div div{
...
}
```

Aplicar las reglas implica reglas en cascada

Los navegadores utilizan la siguiente jerarquia para resentar las hojas de estilo:

- las hojas de estilo predeterminadas del navegador
- las hojas de estilo proporcionadas por el autor de la página
- las hojas de estilo del usuario.

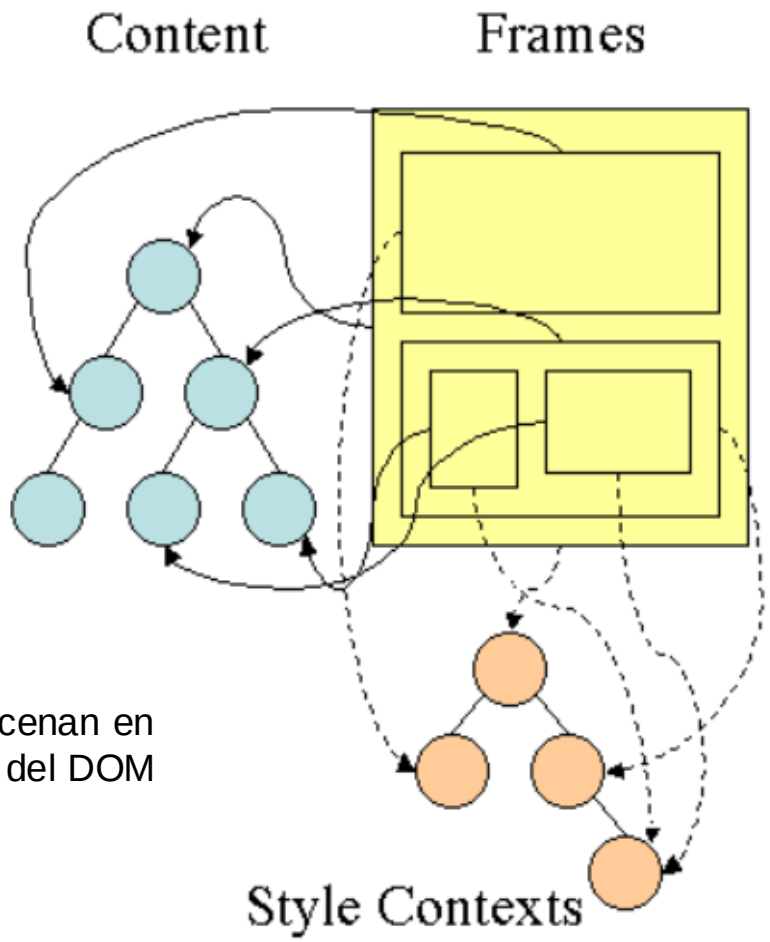
El motor de renderizado

Cálculo de estilo:

Firefox tiene dos árboles adicionales para facilitar el cálculo del estilo: el árbol de reglas y el árbol de contexto de estilo.

Los valores se calculan aplicando todas las reglas de coincidencia en el orden correcto y realizando manipulaciones que los transformen de valores lógicos a concretos.

Por ejemplo, si el valor lógico es un porcentaje de la pantalla, se calculará y transformará en unidades absolutas.



WebKit también tiene objetos de estilo, pero no se almacenan en un árbol como el árbol de contexto de estilo; solo el nodo del DOM señala su estilo correspondiente.

El motor de renderizado

Cálculo de estilo:

Los contextos de diseño se dividen en **structs**. Estos structs contienen información de estilo para una categoría determinada, como borde o color.

El árbol nos ayuda a almacenar en caché structs completos (que contienen los valores finales calculados) en el árbol.

La idea es que, si el nodo inferior no proporciona una definición de struct, se puede usar un struct almacenado en caché en un nodo superior.

Comenzamos en el nodo inferior de la ruta de acceso, el que tiene la prioridad más alta (por lo general, el selector más específico) y atravesamos el árbol hasta que nuestro struct esté completo.

Si no encontramos ninguna definición para nuestro struct, entonces, en caso de que el struct sea un "heredado". apuntamos a la struct de nuestro elemento superior en el árbol de contexto

Visor de arbol DOM: https://software.hixie.ch/utilities/js/live-dom-viewer/

El motor de renderizado

```
<html>
  <body>
    <div class="err" id="div1">
      <p>
        this is a <span class="big"> big error </span>
        this is also a
        <span class="big"> very  big  error</span> error
      </p>
    </div>
    <div class="err" id="div2">another error</div>
  </body>
</html>
```

```
div {margin: 5px; color:black}
.err {color:red}
.big {margin-top:3px}
div span {margin-bottom:4px}
#div1 {color:blue}
#div2 {color:green}
```

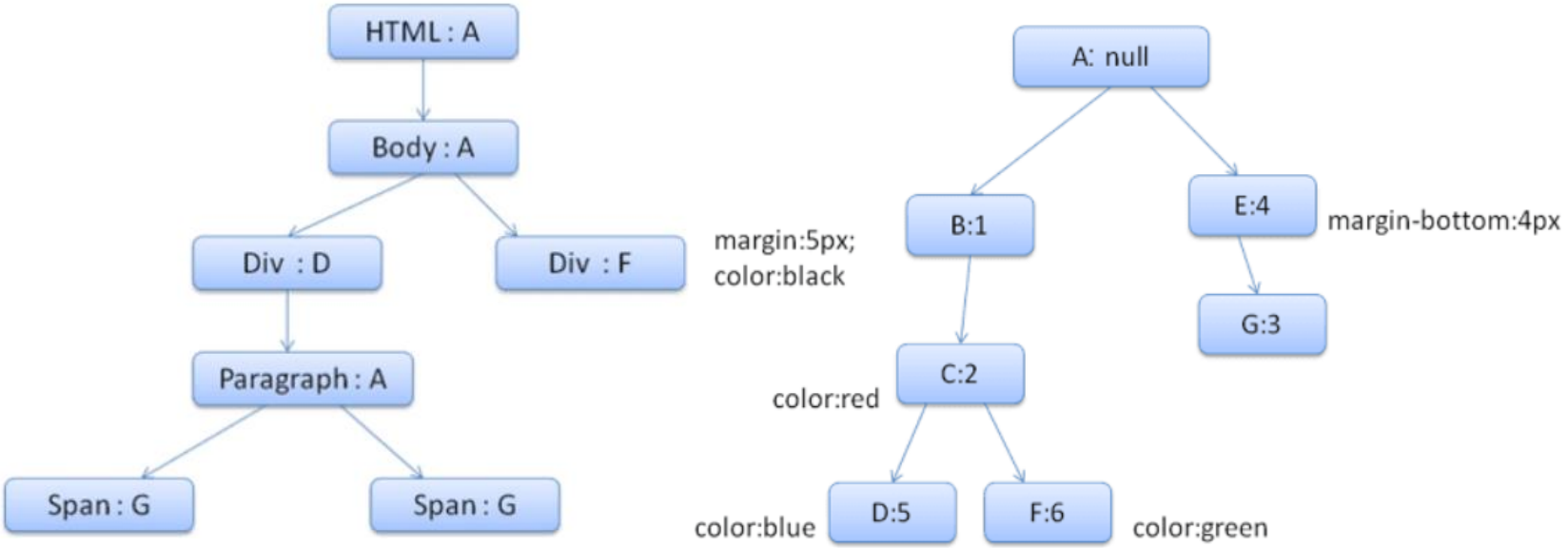


Figura 16: El árbol de reglas

El motor de renderizado

El lienzo

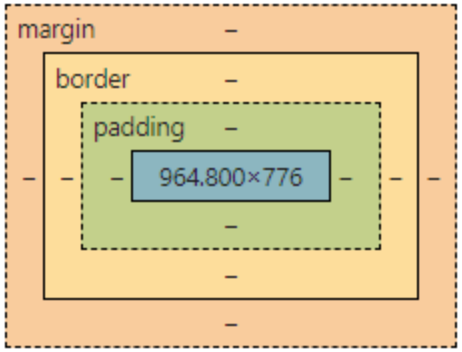
El término lienzo describe "el espacio donde se renderiza la estructura de formato": donde el navegador pinta el contenido.

El lienzo es infinito para cada dimensión del espacio, pero los navegadores eligen un ancho inicial en función de las dimensiones del viewport.

Según www.w3.org/TR/CSS2/zindex.html, El lienzo es transparente si está contenido dentro de otro y, de lo contrario, se le da un color definido por el navegador.

El modelo de cuadro CSS

El modelo de cuadro CSS describe los cuadros rectangulares que se generan para los elementos del árbol de documentos y que se disponen según el modelo de formato visual.



El motor de renderizado

Painting

Después de la construcción del árbol de renderización, se pasa por un “diseño”. el proceso de administración de recursos. Esto significa que debes indicar a cada nodo las coordenadas exactas del lugar en el que debe aparecer en la pantalla. La siguiente etapa es Painting: Se recorrerá el árbol de renderización y se pintará cada nodo con la capa de backend de la IU.

El orden de pintado: Este es, en realidad, el orden en que se apilan los elementos en los contextos de pila. Este orden afecta a la pintura, ya que las pilas se pintan de atrás hacia adelante. El orden de apilado de un procesador de bloques es el siguiente:

- background color
- imagen de fondo
- borde
- niños
- descripción

Componentes principales

El resto de componentes principales forman parte del resto de las asignaturas:

- **Herramientas de redes:** Para llamadas de red, como solicitudes HTTP, mediante diferentes implementaciones para diferentes plataformas detrás de una interfaz independiente de la plataforma.
- **Backend de la IU:** Se usa para dibujar widgets básicos, como ventanas y cuadros combinados. Este backend expone una interfaz genérica que no es específica de la plataforma. Debajo, se usan los métodos de la interfaz de usuario del sistema operativo.
- **Intérprete de JavaScript.** Se usa para analizar y ejecutar código JavaScript.
- **Almacenamiento de datos.** Esta es una capa de persistencia. Es posible que el navegador deba guardar todo tipo de datos de forma local, como las cookies. Los navegadores también admiten mecanismos de almacenamiento como localStorage, IndexedDB, WebSQL y FileSystem.