¿Cómo funciona Mozilla Firefox? Motor Gecko y el Proyecto Quantum

Interesante es saber sobre cómo funciona el navegador web Mozilla Firefox (para nosotros, nuestro favorito) y vamos a diseccionar y explicar de manera sencilla, sin mayores pretensiones, sus diversos componentes. Para ello comenzaremos de atrás para adelante: el último Proyecto de Mozilla Firefox llamado "**Quantum**", ¡Investiguemos juntos!

Introducción.

"**Quantum**" es, como dijimos, el último proyecto que hace de Mozilla Firefox 53 una versión destacada sobre las anteriores. Tan grande es el proyecto que hubo la necesidad de ir abonando el camino para ello con el Proyecto "Electrolysis" apuntando así a un objetivo más alto. Nosotros, por estos lares, hemos seguido de cerca la evolución de los procesadores de 32 bits a 64 bits y ha sido nuestro mayor "trauma" a la hora de programar y compilar en diferentes ambientes. **Pero he aquí que en los programadores de Mozilla están más bien preocupados por los dispositivos con múltiples procesadores y, de paso, en diferentes hardwares (ordenadores de mesa, dispositivos portátiles, móviles celulares y paren ustedes de contar).**

Ellos primero están apostando a los procesos paralelos: cada pestaña o ventana es un hilo de programación y si colapsa solo reiniciamos el que falló, **no todo el proceso principal de Firefox.** Por supuesto, no solo esto lo han mejorado, nosotros por acá hemos explicado la novedosa Norma de Diseño de Cuadrícula sin contar que incluyeron temas "compactos", nuevas extensiones para el navegador, "máscaras" de CSS, mejor experiencia con archivos multimedia, y los usuarios en Microsoft Windows pueden seleccionar si instalan la versión de 32 ó 64 bits.

Componentes principales de Gecko.

Para finalizar con la introducción diremos que muchos de los componentes del proyecto "**Quantum**" están desarrollados en lenguaje **Rust**, solo por mencionar ya que no profundizaremos en el asunto (abajo están los enlaces si queréis ir más lejos). Los componentes a la fecha son los siguientes:

- **rust-bindgen**: es el "conector" entre el lenguaje C++ {con el que está(ba) escrito el código de Firefox} y el lenguaje Rust.
- **Quantum CSS 'Stylo'**: maneja las Hojas de Estilo en Cascada pero de manera "paralela".
- **Quantum Render**: se encarga de "dibujar" en pantalla.
- **Quantum DOM:** se encarga de estructurar el archivo recibido en HTML.
- **Quantum Flow**: para analizar el rendimiento y para mejorar la aplicación.

Más adelante explicaremos los componentes considerados básicos, mantendremos al mínimo este tutorial.

Nociones básicas.

Un navegador web es un software que lee un archivo escrito en lenguaje HTML y lo interpreta según normas recomendadas y preestablecidas para mostrarlo a nosotros de una manera interactiva. Dicho archivo reposa en un servidor web *y asumiremos que simplemente es un fichero que reposa en dicho sitio* -esto no es rigurosamente cierto ya que hoy en día las págnas web son dinámicas, es decir se presentan de diferentes maneras según el navegador, usuario, datos solicitados e incluso según la ubicación y dispositivo-.

Decimos que un *navegador web* tiene un **motor web ("browser engine")** y en el caso de Firefox este motor es llamado **Gecko** el cual tiene muchos componentes de lo cuales nombramos algunos en la introducción.

Aclaratoria necesaria.

Todo lo que aquí presentamos es cierto para la versión Firefox 53 el cual esperamos tengáis

instalado ya que hay varios elementos gráficos que solo se mostrarán correctamente en esa versión (o superior).

Simplificando al máximo.

Haciendo un panorama general, toda página web se puede dividir en tres partes principales:

- El código que representa la estructura de la página (HTML).
- El código que añade un estilo a la estructura anterior (CSS).
- El código que añade interactividad con el usuario y que puede modificar los dos anteriores para así repetir el ciclo sin necesidad de descargar de nuevo la página.

Documento HTML (estructura).

Como establecimos y es un hecho cierto al nosotros ejecutar cualquier navegador web y solicitar en la barra de direcciones un dominio cualquiera lo que recibimos *es simplemente un fichero con instrucciones en HTML*. **Un vez recibido es cuando realmente comienza el proceso de leer, analizar e interpretar dicho código.** En inglés a esta pieza de software le llaman **parser** por su derivación del idioma latín "***pars orationis"*** *(parte del discurso, estructura gramatical)* y que ellos simplemente abrevian **pars**, lo toman como si fuera un verbo y lo convierten en sustantivo agregandole el sufijo "**er":** es así que tenemos el **parser** y que nosotros lo llamaremos **analizador** (ya la web abunda en inglés, debemos construirla también en castellano ¡ayúdanos siguiendo el ejemplo!).

Modelo de Objeto Documento («Document Object Model 'DOM'»).

Acá es donde interviene el primer **analizador**: se encarga de categorizar y formar un árbol a partir de las etiquetas claves en HTML: **head, body, div, footer** y va más allá con cada uno de los párrafos, titulares, etiquetas personalizadas, etc. de cada sección. Lo más importante es que forma un índice de elementos a modo de catálogo para que cuando sea consultado por medio de otros componentes pueda ser retribuido rápidamente. Recordad también que debido al JavaScript dicho DOM puede ser modificado posteriormente y debe ser capaz de registrar y llevar control de dichos cambios.

Nosotros vemos al DOM como una especie de base de datos donde almacenamos el código HTML de una manera normalizada.

Hojas de Estilo en Cascada («Cascading Style Sheets 'CSS'»).

Ahora tenemos otro analizador: el que se encarga de darle formato al código HTML que está categorizado en el DOM. *En teoría cada elemento en el CSS debería tener su correspondiente en*

el DOM. De igual manera este analizador se encarga de categorizar de manera sucesiva los estilos que estén en el fichero -o sección- CSS. Decimos de manera sucesiva porque tal vez se nombre varias veces al mismo elemento HTML y el último en ser leído será el estilo como será presentado (esto no sucede a menudo por ser un desperdicio de trabajo redundar en el estilizado de componentes). Este último resultado es el que se va a proceder a enviar al motor de diseño.

Motor de Diseño («Layout»).

Ya con el estilo aplicado cada uno de los componentes son preparados para ser dibujados y dejarlo listo en capas para pasarlo al siguiente paso.

Motor de Capas

Recibe las capas y las lleva en orden de jerarquía según se haya determinado y las mantiene ordenadas y listas para ser presentadas al motor de dibujo.

Motor de Dibujo.

Acá se coordina todo dejando los espacios necesarios entre cada capa (que se superpone y que se puede ver y que se oculta) además de estar pendiente de las posibles modificaciones a futuro producidas por el JavaScript.

Motor "Compositor".

Recibe y abre en procesos paralelos cada página web que visualize el usuario, ya sea en una pestaña o una ventana nueva y si esto falla por alguna razón pues entonces vuelve a abrir otro proceso solicitando de nuevo al motor de dibujo que se mantiene incólume corriendo en un proceso aparte.

Motor de JavaScript.

Acá se establecen los eventos que disparará el usuario y que podrá modificar tanto como el HTML, el CSS o ambos al mismo tiempo (la mayoría de las veces el proceso no es tan "traumático", podría ser simplemente mostrar un texto distinto lo cual es algo más sencillo de realizar por parte del sistema operativo que por la maquinaría de Firefox en sí).

Conclusiones.

Fuentes consultadas:

En idioma inglés:

- «What is a browser engine? ? Mozilla Hacks – the Web developer blog» by Potch.
- «A Quantum Leap for the Web» by Dvaid Bryant.
- «GPU Process in Beta 53» by Anthony Hughes.
- «Rust's getting started».
- «Firefox 53: Quantum Compositor, Compact Themes, CSS Masks, and More» by Dan Callahan.
- «Official Quantum's Wiki».
- «How Browsers Work: Behind the scenes of modern web browsers» by By Tali Garsiel and Paul Irish.
- «Built for those who build the Web» Mozilla Developer.

En idioma italiano:

- «Arriva FIREFOX 19.0 con lettore PDF integrato».