

¿Cómo funciona Internet?

Introducción a Internet

Red (Network): Es un grupo de computadoras u otros dispositivos que están conectados entre sí.

Cuando muchas redes se conectan entre sí, forman el Internet.

"Internet es una red de redes"

Cómo funciona Internet: Descripción General

En un nivel alto, Internet funciona conectando dispositivos y sistemas informáticos mediante un conjunto de protocolos estandarizadas.

Estos protocolos definen cómo se intercambian la info entre dispositivos y garantizan que los datos se transmitan de manera confiable y segura.

El núcleo de Internet es una red global de Router interconectados, que dirigen el tráfico entre diferentes dispositivos y sistemas. Cuando envío datos a través de Internet, se dividen en pequeños paquetes.

El Router examina el paquete y lo reenvía al siguiente Router en el camino hacia su destino.

Para garantizar que los paquetes se envíen y reciben correctamente, existen una variedad de protocolos, incluido IP y TCP.

IP: Responsable de enrutar los paquetes a su destino correcto

TCP: Garantiza que los paquetes se reciban de manera confiable y en el orden correcto.

Además de estos protocolos básicos, existe una amplia gama de otras tecnologías y protocolos que se utilizan para permitir la comunicación y el intercambio de datos a través de Internet.

Incluido el DNS, HTTP y SSL/TLS.

Conceptos básicos y Terminología

Paquete: Una unidad pequeña de datos que se transmite a través de Internet.

Router: Dispositivo que dirige paquetes de datos entre diferentes redes.

Dirección IP: Un identificador único asignado a cada dispositivo en una red, usado para enrutar datos a la dirección correcta.

Domain Name (Nombre de dominio): Un nombre legible para humanos que se utiliza para identificar un sitio web, como google.com.

DNS: Domain Name System es responsable de traducir los nombres de dominio en direcciones IP.

HTTP: Hypertext Transfer Protocol es utilizado para transferir datos entre un cliente (navegador web por ej.) y un servidor (ej: sitio web).

HTTPS: Versión encriptada de HTTP utilizada para proveer una comunicación segura entre un cliente y un servidor.

SSL/TLS: Secure Sockets Layer y Transport Layer Security, son protocolos utilizados para proporcionar una comunicación segura a través de Internet.

El Rol de los Protocolos en Internet:

Un protocolo es un conjunto de reglas y estándares que definen cómo se intercambian la información entre dispositivos y sistemas.

Uno de los beneficios clave de utilizar protocolos estandarizados es que permiten que dispositivos y sistemas de diferentes fabricantes y proveedores se comuniquen entre sí sin problemas.

Comprendiendo las direcciones IP y los nombres de dominio

Las direcciones IP son típicamente representadas como una serie de 4 números separados por puntos, como "192.168.1.1".

Los nombres de dominio están típicamente compuestos de dos o más partes, separadas por puntos. Estos son traducidos a direcciones IP utilizando el DNS.

Cuando entras un nombre de dominio en tu navegador web, tu PC envía una petición DNS al servidor DNS, el cual devuelve la dir. IP correspondiente. Finalmente, el PC utiliza esa dir. IP para conectarse a la web/recurso que se ha solicitado.

Introducción a HTTP & HTTPS

HTTP es el protocolo utilizado para transferir datos entre un cliente y un servidor. Cuando visitas una web, tu navegador envía una petición HTTP al servidor, pidiendo la página web u otro recurso que hayas solicitado. El servidor envía una respuesta HTTP de vuelta al cliente, conteniendo los datos solicitados.

HTTPS es una versión más segura de HTTP, el cual cifra los datos que se transmiten entre el cliente y el servidor mediante cifrado SSL/TLS. Esto proporciona una capa adicional de seguridad y ayuda a proteger información confidencial.

Creación de aplicaciones con TCP/IP

Al crear aplicaciones con TCP/IP, hay algunos conceptos clave que se deben comprender:

- **Ports (puertos)**: Los puertos son utilizados para identificar la app o servicio ejecutándose en un dispositivo. Cada app o servicio es asignado a un único número de puerto, permitiendo que los datos seren enviados a la dirección correcta.
- **Sockets**: Un socket es una combinación de un dir. IP y un puerto representando un punto final (endpoint) específico para la comunicación. Se utilizan para establecer conexiones entre dispositivos y transferir datos entre aplicaciones.
- **Conexiones**: Una conexión es establecida entre 2 sockets cuando 2 dispositivos se quieren comunicar entre ellos. Durante el proceso de establecimiento de la conexión, los dispositivos negocian varios parámetros que determinan cómo los datos serán transmitidos a través de la conexión.
- **Datagram Transfer**: Una vez la conexión se ha establecido, los datos pueden ser transferidos entre las aplicaciones ejecutándose en cada dispositivo. Los datos se transmiten en segmentos, y cada segmento contiene un número de secuencia y otras medidas para garantizar una entrega confiable.

Asegurar la comunicación por Internet con SSL/TLS

Cuando se utiliza SSL/TLS para proteger la comunicación por Internet, hay algunos conceptos clave que se deben comprender:

- **Certificados**: Los certificados SSL/TLS son utilizados para establecer confianza entre el cliente y el servidor. Contienen info sobre la identidad del servidor, están firmados por un Tercero de confianza para verificar su autenticidad.

• **Handshake**: Durante el proceso de handshake SSL/TLS, el cliente y el servidor intercambian información para negociar el algoritmo de cifrado y otros parámetros para la conexión segura.

• **Cifrado**: Una vez que se establece la conexión segura, los datos se cifran utilizando el algoritmo acordado y pueden transmitirse de forma segura entre el cliente y el servidor.

¿Qué es HTTP?

HTTP es un protocolo de comunicación de capa de aplicación basado en TCP/IP que especifica cómo los clientes y servidores se comunican entre sí.

¿Qué hay en una petición HTTP?

Una solicitud HTTP es la forma en que las plataformas de comunicaciones de Internet solicitan la información necesaria para cargar un sitio web.

Cada solicitud HTTP lleva consigo una serie de datos codificados que contienen diferentes tipos de información. Una solicitud HTTP típica contiene:

1. Tipo de versión HTTP
2. URL
3. Método HTTP
4. Encabezados de solicitud HTTP
5. Cuerpo HTTP (opcional)

Método HTTP

Un método HTTP indica la acción que la solicitud HTTP espera del servidor consultado.

2 métodos comunes:

- 'GET': Espera info a cambio
- 'POST': Generalmente indica que el cliente está enviando info al servidor web

Encabezados de solicitud HTTP

Los encabezados HTTP contienen información de texto almacenadas en pares clave - valor, y se incluyen en cada petición HTTP (y respuesta). Estos encabezados comunican información básica, como qué navegador es el utilizado el cliente y qué datos se solicitan.

Ejemplo:

```
▼ Request Headers
:authority: www.google.com
:method: GET
:path: /
:scheme: https
accept: text/html
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0
```

Cuerpo de solicitud HTTP

El cuerpo de una petición es la parte que contiene el 'cuerpo' de la información que se transfiere la solicitud. El cuerpo de una petición HTTP contiene cualquier información que se envía al servidor web, como el nombre de usuario y la contraseña, o cualquier otro dato ingresado en un formulario.

¿Qué hay en una respuesta HTTP?

Una respuesta HTTP es lo que los clientes web reciben de un servidor web como respuesta a una petición HTTP. Estas respuestas comunican información válida basada en lo que se solicitó en la petición HTTP.

Una típica respuesta HTTP contiene:

1. Un código de estado HTTP (status code)
2. Encabezados de respuesta HTTP
3. Cuerpo HTTP (opcional)

HTTP status codes

Los códigos de estado HTTP son códigos de 3 dígitos que se utilizan con mayor frecuencia para indicar si una solicitud HTTP se completó correctamente.

- **1xx**: Información
- **2xx**: Éxito
- **3xx**: Redirección
- **4xx**: Error del cliente
- **5xx**: Error del servidor

Encabezados de respuesta HTTP

A igual que una petición HTTP, una respuesta HTTP viene con encabezados que transmiten info importante, como el idioma y el formato de los datos que se envían en el cuerpo de la respuesta.

Ejemplo:

```
▼ Response Headers
cache-control: private, max-age=0
content-encoding: br
content-type: text/html; charset=UTF-8
date: Thu, 21 Dec 2017 18:25:08 GMT
status: 200
strict-transport-security: max-age=86400
x-frame-options: SAMEORIGIN
```

Cuerpo de respuesta HTTP

Las respuestas HTTP exitosas a solicitudes 'GET' generalmente tienen un cuerpo que contiene la info solicitada. En la mayoría de peticiones web, se tratará de datos HTML que un navegador web traducirá a una pg web.

¿Cómo funcionan los navegadores?

La función principal de los navegadores:

La función principal de un navegador es presentar el recurso web que escoges, solicitándolo al servidor y mostrándolo en la ventana del navegador.

El recurso puede ser un documento HTML, pero también puede ser un PDF, una imagen, etc.

La ubicación del recurso la especifica el usuario mediante una URI (Uniform Resource Identifier).

La forma en que el navegador interpreta y muestra los archivos HTML se especifica en las especificaciones HTML y CSS.

Las interfaces de usuario del navegador tienen mucho en común entre sí. Entre los elementos comunes de la interfaz de usuario se encuentran:

- ① Barra de direcciones para insertar una URL
- ② Botones de avance y retroceso
- ③ Opciones de marcadores
- ④ Botones Actualizar y Detener
- ⑤ Botón de inicio

La estructura de alto nivel del navegador:

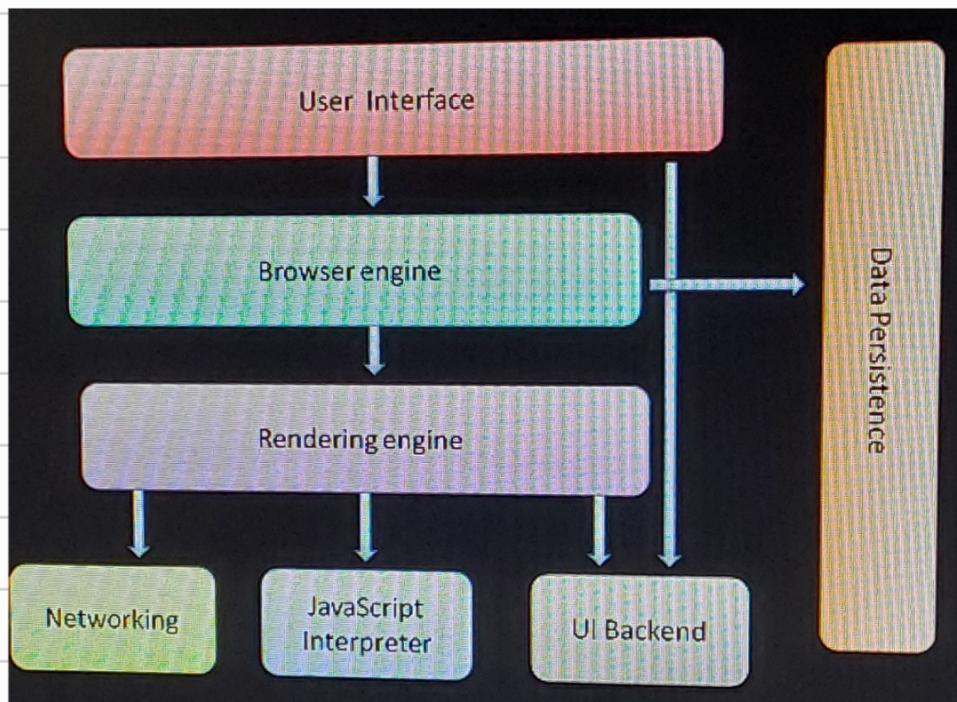
Los principales componentes del navegador son:

- 1 La interfaz de usuario
- 2 El motor del navegador: Organiza acciones entre la interfaz de usuario y el motor de renderizado.
- 3 El motor de renderizado: Responsable de mostrar el contenido solicitado.
- 4 Networking: Para mandar de red, como solicitudes HTTP, utilizando diferentes implementaciones para diferentes plataformas detrás de una interfaz independiente de la plataforma.

5 UI Backend: Se utiliza para dibujar widgets básicos como cuadros combinatorios y ventanas. Este backend expone una interfaz genérica que no es específica de la plataforma. Debajo utilizan todos los métodos de interfaz de usuario del SO.

6 Interpretador de JavaScript: Se utiliza para analizar y ejecutar código JavaScript.

7 Almacenamiento de datos: Capa de persistencia. Es posible que el navegador necesite guardar todo tipo de datos localmente con cookies.



El motor de renderizado

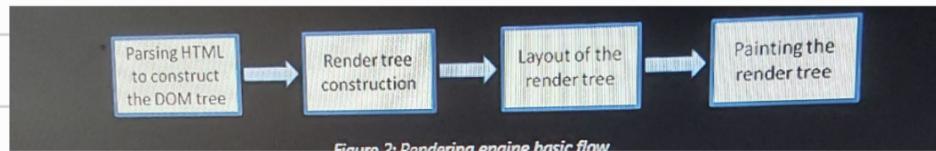
Su responsabilidad es el renderizado, la visualización del contenido solicitado en la pantalla del navegador.

De forma predeterminada, el motor de renderizado puede mostrar documentos e imágenes HTML y XML. También otros tipos de datos mediante extensiones o complementos (PDF con visor de PDF).

El flujo principal

El motor de renderizado comienza a obtener el contenido del documento solicitado desde la capa de red.

Después de eso, este es el flujo básico del motor de renderizado:



El motor comenzará a analizar el documento HTML y convertir sus elementos en nodos DOM en un árbol llamado "árbol de contenido". El motor analizará los datos de estilo. La información de estilo junta con instrucciones visuales en HTML se utilizarán para crear otro árbol: el árbol de renderizado.

El árbol de renderizado contiene rectángulos con atributos visuales como color y dimensiones. Están en el orden correcto para mostrarse en la pantalla.

Después de la construcción del árbol de renderizado, pasa por un proceso de "diseño (layout)". Esto significa darle a cada nodo las coordenadas exactas donde debería aparecer en la pantalla.

La siguiente etapa es pintar; se recorrerá el árbol de renderizado y cada nodo se pintará usando la capa backend de la interfaz de usuario.

Es importante entender que este es un proceso gradual. Para una mejor experiencia del usuario, el motor de renderizado intentará mostrar el contenido en la pantalla lo antes posible. Partes del contenido se analizan y muestran, mientras el proceso continúa con el resto de los contenidos que siguen llegando de la red.

Parsing - general

Analizar (parsing) un documento significa traducirlo a una estructura que el código pueda usar. El resultado del análisis suele ser un árbol de nodos que representan la estructura del documento.

Es lo que se llama **árbol de análisis** (parse tree) o **árbol de sintaxis** (syntax tree).

Por ejemplo, al analizar la expresión "2 + 3 - 1" se podría devolver este árbol:

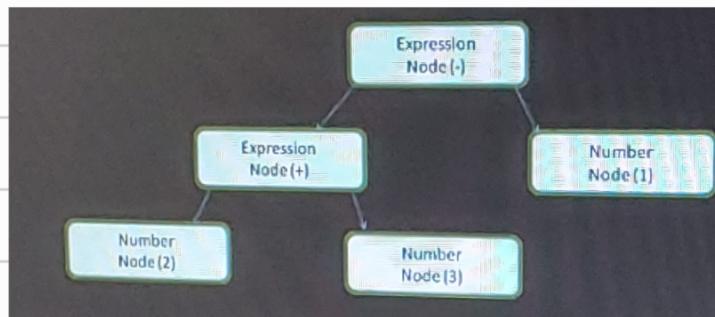


Figura 5

Gramáticas

El análisis se basa en las reglas de sintaxis que obedece el documento: el idioma o formato en el que fue escrito. Cada formato que pueda analizar debe tener una gramática determinista que consiste en vocabulario y reglas de sintaxis. Se llama **gramática libre de contexto**.

Parser - Lexer Combinación

El análisis se puede dividir en dos subprocesos

→ Análisis Léxico

→ Análisis de sintaxis

Análisis Léxico → Proceso de dividir la entrada en tokens. Los tokens son el vocabulario del lenguaje: la colección de bloques de construcción válidos.

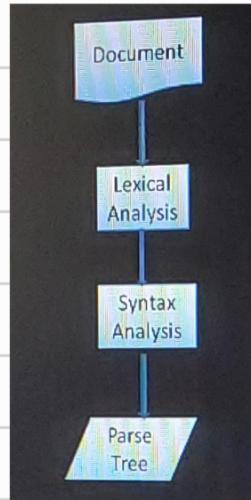
Análisis de sintaxis → Aplicación de las reglas de sintaxis del lenguaje

Los analizadores (parsers) generalmente dividen el trabajo entre dos componentes:

- el lexer (a veces llamado Tokenizer), responsable de dividir la entrada en tokens válidos.

- el organizador, responsable de construir el árbol de análisis analizando la estructura del doc de acuerdo con las reglas de sintaxis del lenguaje.

El lexer sabe cómo eliminar caracteres irrelevantes como espacios en blanco y saltos de líneas.



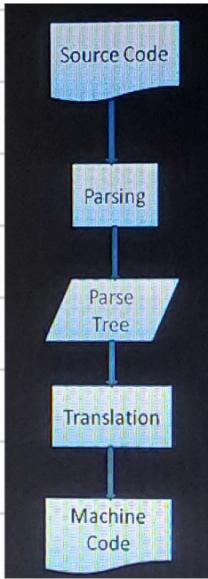
El proceso de análisis es iterativo. El analizador normalmente le pedirá al lexer un nuevo Token e intentará hacer coincidir el Token con una de las reglas de sintaxis. Si una regla coincide, se agregará un nodo correspondiente al Token al árbol de análisis y el analizador solicitará otro Token.

Si ninguna regla coincide, el analizador almacenará el Token internamente y seguirá solicitando Tokens hasta que encuentre una regla que coincide con todos los Tokens almacenados internamente. Si no se encuentra ninguna regla, el analizador generará una excepción. Esto significa que el documento no era válido y contenía errores de sintaxis.

Traducción

En muchos casos, el árbol de análisis no es el producto final. El análisis se utiliza a menudo en la Traducción o Transformar el documento de entrada a otro formato. Un ejemplo es la compilación.

El compilador que compila el código fuente en código de máquina primero lo analiza en un árbol de análisis y luego traduce el árbol en un documento de código de máquina.



Ejemplo de análisis.

En la figura 5 construimos un árbol de análisis a partir de una expresión matemática (véase pág 16). Intentemos definir un lenguaje matemático simple y veámos al proceso de análisis.

Sintaxis

- ① Los componentes básicos de la sintaxis del lenguaje son expresiones, términos y operaciones.
- ② Nuestro lenguaje puede incluir cualquier número de expresiones.
- ③ Una expresión se define como "Término" seguido de una "Operación" seguida de otro Término.
- ④ Una operación es un Token positivo o negativo.
- ⑤ Un Término es un Token entero (integer) o una expresión.

Analicemos la entrada $2 + 3 - 1$.

La 1a subcadena que coincide con una regla es 2 : según la regla #5 es un Término. La 2a coincidencia es $2 + 3$: coincide con la regla #3. La siguiente coincidencia solo se alcanzará al final de la entrada.

$2 + 3 - 1$ es una expresión porque ya sabemos que $2 + 3$ es un Término, entonces tenemos un Término seguido de una operación seguida de otro Término, $2 + t$ no coincide con ninguna regla y por lo tanto es una entrada no válida.

Definiciones formales de vocabulario y sintaxis

El vocabulario suele expresarse mediante expresiones regulares.

Una expresión regular (regex o regexp) es una cadena de texto especial para describir un patrón de búsqueda. Probablemente esté familiarizado con las notaciones comodín (wildcard notations), como `*`, `?`, para buscar todos los archivos de texto en un administrador de archivos.

El equivalente de expresión regular es `^.*\.txt$`

Por ejemplo nuestro lenguaje se definirá como:

```
INTEGER: 0|[1-9][0-9]*
PLUS: +
MINUS: -
```

Como puede ver, los números enteros se definen mediante una expresión regular. La sintaxis suele definirse en un formato llamado BNF. Nuestro idioma se definirá como:

```
expression ::= term operation term
operation ::= PLUS | MINUS
term ::= INTEGER | expression
```

Tipos de analizadores

Hay 2 tipos:

- **Analizadores de arriba hacia abajo** → Examinan la estructura de alto nivel de la sintaxis e intentan encontrar una coincidencia de reglas.
- **Analizadores de abajo hacia arriba** → Comienzan con la entrada y la transforman gradualmente en reglas de sintaxis, comenzando desde las reglas de bajo nivel hasta que se cumplen las reglas de alto nivel.

Veamos cómo los 2 tipos analizan nuestro ejemplo:

El analizador de arriba hacia abajo comenzará desde la regla de nivel superior e identificará $2+3$ como una expresión. Luego identificará $2+3-1$ como una expresión (el proceso de identificación de la expresión evoluciona, coincidiendo con las otras reglas, pero el punto de partida es la regla de nivel más alto).

El analizador de abajo hacia arriba examinará la entrada hasta que coincida una regla. Luego reemplazará la entrada coincidente con la regla. Esto continuará hasta el final de la entrada. La expresión parcialmente coincidente se coloca en la pila del analizador.



Stack	Input
	$2 + 3 - 1$
term	$+ 3 - 1$
term operation	$3 - 1$
expression	$- 1$
expression operation	1
expression	-

Este tipo de analizador se maneja analizadores de desplazamiento-reducción porque la entrada se desplaza hacia la derecha y se reduce gradualmente a reglas de síntesis.

Analizador HTML

El trabajo del analizador HTML es analizar el HTML markup en un árbol de análisis.

La definición de gramática HTML

El vocabulario y la sintaxis de HTML están definidos en especificaciones creadas por la organización W3C.

No es una gramática libre de contexto

Desafortunadamente, todos los lenguajes del analizador convencional no se aplican a HTML. HTML no se puede fácilmente mediante una gramática libre de contexto que necesitan los analizadores.

Existe un formato formal para definir HTML DTD (Document Type Definition), pero no es una gramática libre de contexto.

El enfoque HTML es más "indulgente": le permite omitir ciertas etiquetas o, a veces, omitir etiquetas (tags) de inicio o fin, etc. En general, es una sintaxis "suave", a diferencia de la sintaxis rígida y exigente de XML.

HTML DTD

Este formato (DTD) se utiliza para definir versiones de la familia SGML (Standard Generalized Markup Language).

SGML es un estándar para definir lenguajes de marcado (markup) generalizados para documentos.

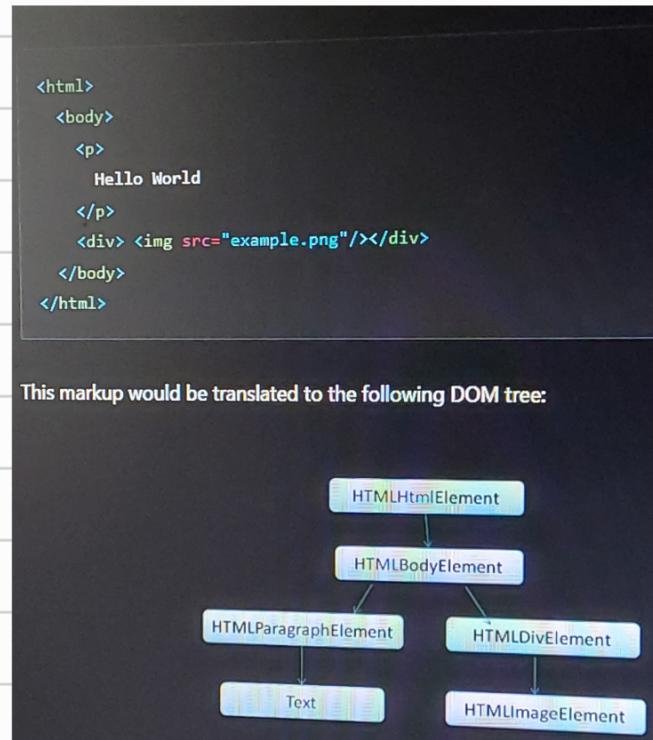
El formato (DTD) contiene definiciones para todos los elementos permitidos, sus atributos y jerarquía.

DOM:

El árbol de salida (el "árbol de análisis") es un árbol de elementos DOM y nodos de escrituras. DOM es la abreviatura de Modelo de objetos de documento (Document Object Model). Es la presentación del objeto del documento HTML y la interfaz de los elementos HTML con el mundo exterior como JavaScript.

La raíz del árbol es el objeto "Documento". La interfaz Documento representa el documento HTML o XML completo. Conceptualmente, es la raíz del árbol del documento y proporciona el acceso principal a los datos del doc.

El DOM tiene una relación casi uno a uno con el markup. Por ejemplo:



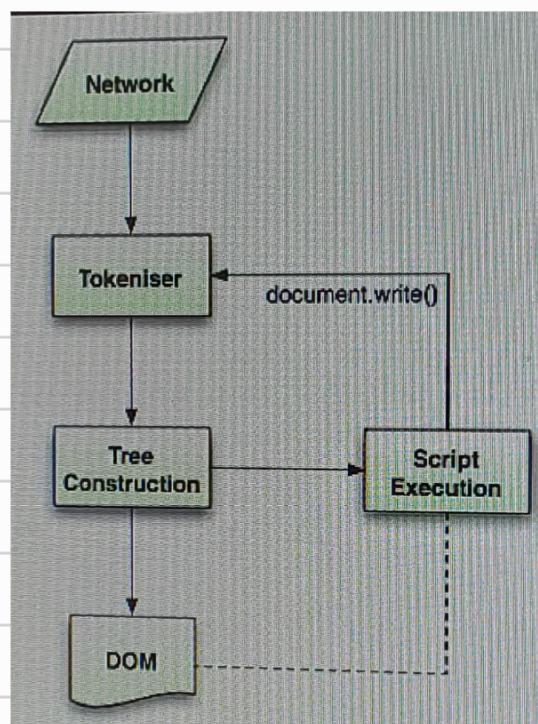
El algoritmo de análisis:

Como vimos en las secciones anteriores, HTML no se puede analizar utilizando los analizadores habituales. Las razones son:

- [1] La naturaleza indulgente del lenguaje
- [2] El hecho de que los navegadores tengan la tradicional tolerancia a errores para admitir casos bien conocidos de HTML no válido.
- [3] El proceso de análisis es reentrante. Para otros lenguajes, da igual no calcular durante el análisis, pero en HTML, el código dinámico puede agregar tokens adicionales, por lo que el proceso de análisis en realidad modifica la entrada.

Los navegadores crean analizadores personalizados para analizar HTML. El algoritmo de análisis consta de 2 etapas:

- **Tokenización:** Es el análisis léxico, que analiza la entrada en tokens. Entre los tokens HTML, se encuentran los tags de inicio, los tags de finalización, los nombres de atributos y sus valores.
- **Tokenizador:** Reconoce el token, se lo entrega al constructor del árbol y consume el siguiente carácter para reconocer el siguiente token, y así sucesivamente hasta el final de la entrada.



El algoritmo de Tokenización:

La salida del algoritmo es un Token HTML. El algoritmo se expresa como una máquina de estados. Cada estado consume uno o más caracteres del flujo de entrada y actualiza el siguiente estado de acuerdo con esos caracteres.

La decisión está influenciada por el estado actual de Tokenización y por el estado de construcción del árbol. Esto significa que el mismo carácter consumido producirá resultados diferentes para el siguiente estado corredor, dependiendo del estado actual.

Ejemplo básico - Tokenización del siguiente HTML:

```
<html>
  <body>
    Hello world
  </body>
</html>
```

El estado inicial es el "Estado de **des**". Cuando se encuentra el carácter **<**, el estado cambia a "Estado de **siguiente abierto**". El consumo de un carácter **a** provoca la creación de un "Token de carácter de **inicio**", el estado cambia a "Estado de **nombre de siguiente**".

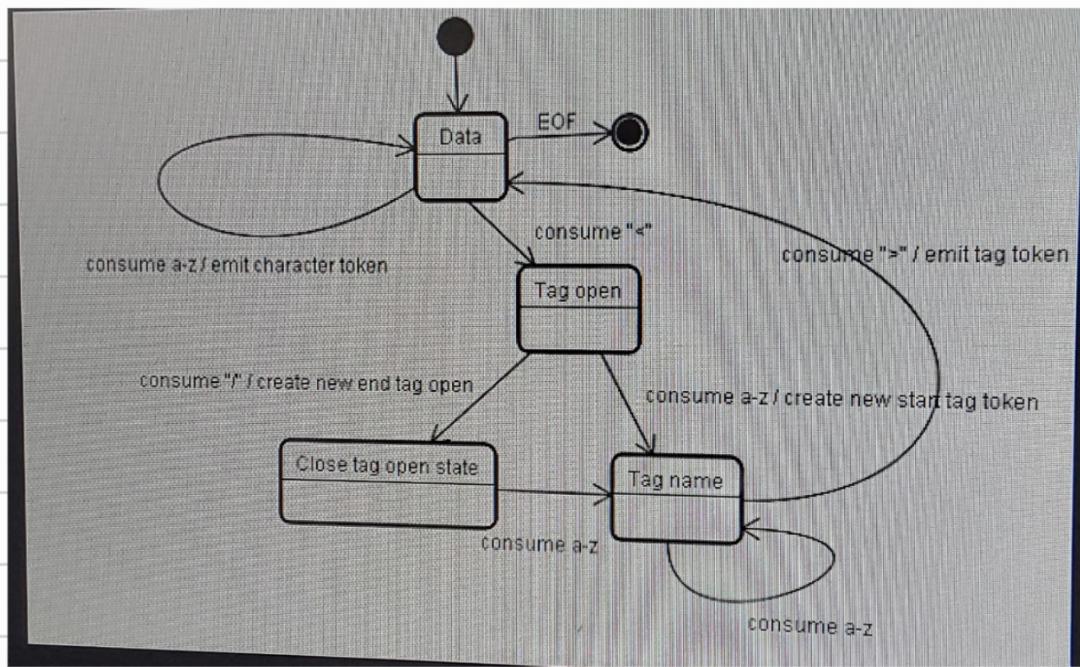
Permanecemos en este estado hasta que se consume el carácter **>**. Cada carácter se agrega al nuevo nombre del Token. En nuestro caso el Token creado es un Token HTML.

Cuando se alcanza la etiqueta **>**, se emite el Token actual y el estado vuelve a ser "Estado de **des**". La etiqueta **<body>** se tratará con los mismos pasos.

Ahora volvemos al "Estado de **des**". Consumir el carácter **H** de **Mola Mundo** provocará la creación y emisión de un Token de carácter, esto continúa hasta que se alcanza el **<** de **</body>**. Emisaremos un Token de carácter por cada carácter de **Mola Mundo**.

Ahora volvemos al "Estado de **siguiente abierto**". Consumir la siguiente entrada / provocará la creación de un Token de **siguiente final** y un cambio al "Estado de **nombre de siguiente**".

Nuevamente permanecemos en este estado hasta llegar a >. Luego se emite el nuevo Token de etiqueta y volvemos al "Estado de datos". La entrada </html> se tratará como en el caso anterior.



Algoritmo de construcción de árboles

Cuando se crea el tokenizador, se crea el objeto Documento. Durante la etapa de construcción del árbol, se modifica el árbol DOM con el Documento en su raíz y se le agregan elementos. Cada nodo emitido por el Tokenizador será procesado por el constructor del árbol.

Para cada Token, la especificación define qué elemento DOM es relevante para él y se creará para ese Token. El elemento se agrega al árbol DOM y también a la pila de elementos abiertos. Esta pila se utiliza para corregir discrepancias de encadenamiento y etiquetas no cerradas.

El algoritmo también se describe como una máquina de estados. Los estados se denominan "modos de inserción".

Vamos al proceso de construcción del árbol para la entrada del ejemplo de Hello World.

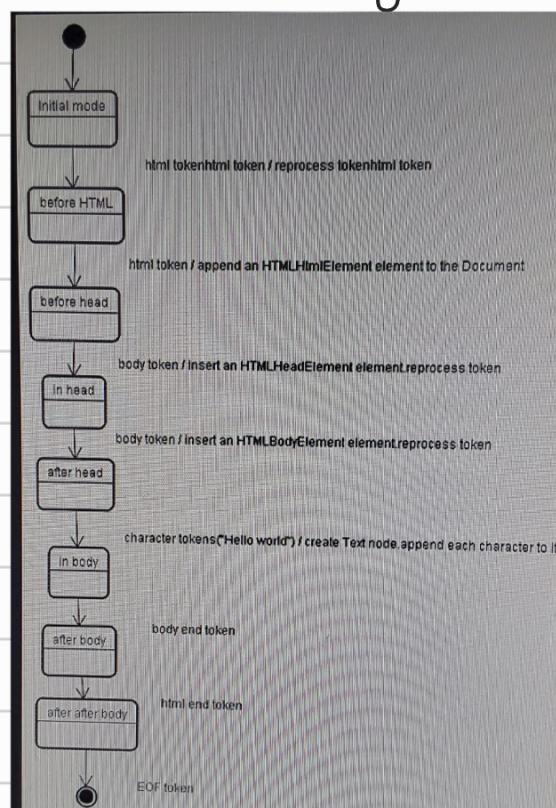
La entrada a la etapa de construcción del árbol es una secuencia de tokens de la etapa de tokenización. El 1er modo es el "modo inicial". Recibir el token "html" provocará un cambio al modo "antes de head" y un reprocessamiento del token en ese modo. Esto provocará la creación del elemento `HTMLHtmlElement`, que se agregará al objeto Documento raíz.

El estado se cambiará a "antes de head". Luego se recibe el token "body". Se creará implícitamente un `HTMLHeadElement` aunque no tengamos un token "head" y se agregará al árbol.

Passamos ahora al modo "in head" y luego al "after head". El token del cuerpo se reprocessa, se crea e inserta un `HTMLBodyElement` y el modo se transfiere a "in body".

Ahora se reciben los tokens de caracteres de la cadena "Hello mundo". El primero provocará la creación e inserción de un nodo "Text" y los demás caracteres se agregarán a ese nodo.

La recepción del token de fin del cuerpo provocará una transferencia al modo "after body". Ahora recibiremos la siguiente final html que nos moverá al modo "after after body". Recibir el token de fin de archivo finalizará el análisis.



Acciones cuando finaliza el análisis

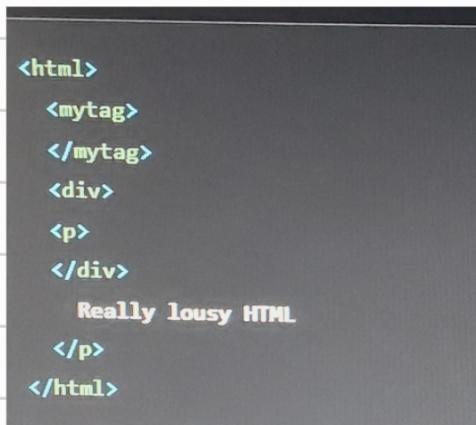
En esta etapa, el navegador marca el documento como interactivo y comienza a realizar los scripts que están en modo "diferido": aquellos que deben ejecutarse después de analizar el documento.

Luego, el estado del documento se establecerá el "completo" y se activará un evento de "load".

Tolerancia a errores de los navegadores

Nunca aparece el error "Invalid Syntax" en una página HTML. Los navegadores corren cualquier contenido no válido y continúan.

Tome este HTML por ejemplo:



```
<html>
<mytag>
</mytag>
<div>
<p>
</div>
Really lousy HTML
</p>
</html>
```

"mytag" no es una etiqueta estandar, sin embargo incosidero, etc. Pero el navegador aún lo muestra correctamente y no se queja. Por lo tanto, gran parte del código del analizador corrige los errores del autor del HTML.

La especificación HTML5 define algunos de estos requisitos. El analizador analiza la estructura tokenizada en el documento, creando el árbol del documento. Si el documento está bien formado, analizarlo es sencillo.

Desafortunadamente, tienen que manejar muchos documentos HTML que no están bien formados, por lo que el analizador debe ser tolerante con los errores.

Tenemos que cuidar al menos las siguientes condiciones de errores

- ① El elemento que se agrega está explícitamente prohibido dentro de alguna etiqueta externa. En este caso debemos cerrar Todas las etiquetas hasta la que prohíbe el elemento y agregarla después.
- ② No se nos permite agregar el elemento directamente. Podría ser que la persona que escribe el documento haya olvidado alguna etiqueta intermedia.
- ③ Queremos agregar un elemento de bloque dentro de un elemento en línea. Cierra Todas las elementos en línea hasta el siguiente elemento de bloque superior.
- ④ Si esto no quita, cierra los elementos hasta que se nos permita agregar el documento, o ignore la etiqueta.

Análisis CSS

A diferencia de HTML, CSS es una gramática libre de contexto y se puede analizar utilizando los tipos de analizadores descritos en la introducción. Veamos algunas ejemplos:

La gramática léxica (vocabulario) se define mediante expresiones regulares para cada token:

```
comment  /\*\[^*]*\*/+([/*][^*]*\*/)*\/
num      [0-9]+|[0-9]"\."[0-9]+
nonascii [\200-\377]
nmstart [_a-zA-Z]\{nonascii\}\{escape\}
nmchar  [_a-zA-Z-]\{nonascii\}\{escape\}
name    \{nmchar\}+
ident   \{nmstart\}\{nmchar\}*
```

"ident" es la abreviatura de identificador, como el nombre de una clase.

"name" es una identificación de elemento (el que se hace referencia con '#').

La gramática sintáctica se describe en BNF.

```
ruleset
  : selector [ ',' S* selector ]*
    '{' S* declaration [ ';' S* declaration ]* '}' S*
  ;
selector
  : simple_selector [ combinator selector | S+ [ combinator? selector ]? ]?
  ;
simple_selector
  : element_name [ HASH | class | attrib | pseudo ]*
    | [ HASH | class | attrib | pseudo ]+
  ;
class
  : '.' IDENT
  ;
element_name
  : IDENT | '*'
  ;
attrib
  : '[' S* IDENT S* [ '=' | INCLUDES | DASHMATCH ] S*
    [ IDENT | STRING ] S* ']'
  ;
pseudo
  : ':' [ IDENT | FUNCTION S* [ IDENT S* ] ')' ]
```

Explicación:

Un ruleset es esta estructura →

div.error y a.error son selectores.

```
div.error, a.error {  
    color:red;  
    font-weight:bold;  
}
```

La parte dentro de los llaves contiene las reglas que aplica este ruleset. Esta estructura se define formalmente en esta definición:

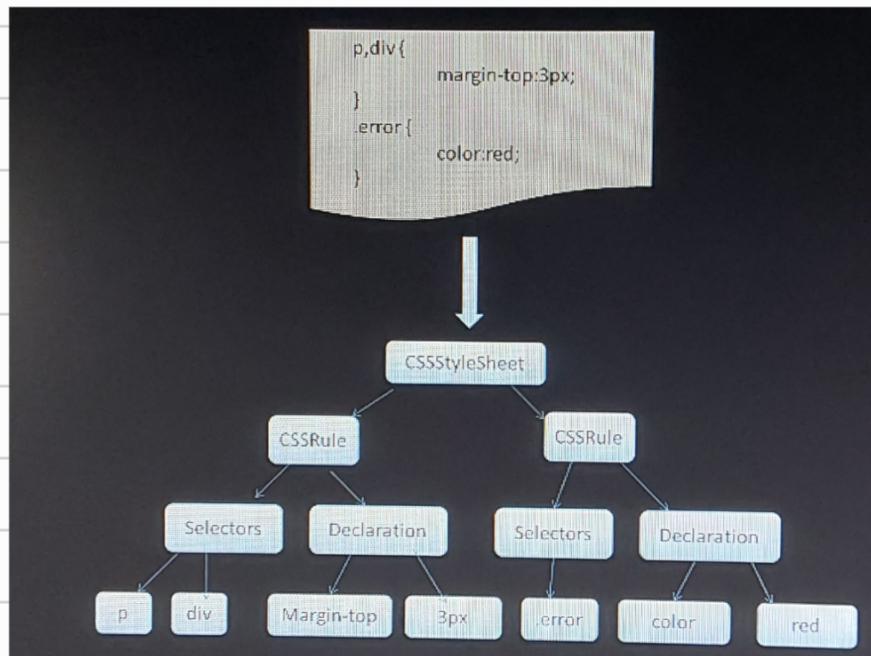
```
ruleset  
: selector [ ',' S* selector ]*  
'{ S* declaration [ ';' S* declaration ]* ' }* S*
```

Esto significa que un ruleset es un selector o, opcionalmente, un número de selectores separados por una coma y espacios (S significa espacio en blanco). Un ruleset contiene llaves y dentro de ellas una declaración o, opcionalmente, un número de declaraciones separadas por un punto y coma.

Analizador CSS WebKit

WebKit utiliza generadores de analizadores Flex y Bison para crear analizadores automáticamente a partir de archivos de gramática CSS. Bison crea un analizador de reducción - desplazamiento.

Firefox utiliza un analizador de arriba hacia abajo escrito manualmente. En ambos casos, cada archivo CSS se traduce en un objeto StyleSheet. Cada objeto contiene reglas CSS. Los objetos de reglas CSS contienen objetos selectores y de declaraciones y otros.



El orden de procesamiento de scripts y hojas de estilo (Style sheets)

Scripts

El modelo de la web es sincrónico. Los autores esperan que los scripts se analicen y ejecuten inmediatamente cuando el analizador alcanza una etiqueta <script>. El análisis del documento se detiene hasta que se ejecuta el script.

Si el script es externo, primero se debe recuperar el recurso de la red; esto también se hace de forma sincrónica y el análisis se detiene hasta que se recupera el recurso. → Modelo duradero

Los autores pueden agregar el atributo "defer" a un script, en cuyo caso no dejará el análisis del documento y se ejecutará después de que se analice el documento. HTML5 agrega una opción para marcar el script como asíncrono para que sea analizado y ejecutado por un hilo diferente.

Análisis especulativo

Tanto Webkit como Firefox realizan este optimización. Mientras se ejecutan los scripts, otro hilo (thread) analiza el resto del documento y descubre qué otros recursos deben cargarse desde la red y los cargar.

De esta forma, se pueden cargar recursos en conexiones paralelas y se mejoran la velocidad general.

NOTA: El analizador especulativo no modifica el árbol DOM.

Style Sheets

Las hojas de estilo tienen un modelo diferente.

Existe el problema de que los scripts solicitan información de estilos durante la etapa de análisis del documento. Si el estilo aún no está cargado y analizado, el script obtendrá respuestas incorrectas y aparentemente esto causa muchos problemas.

Firefox bloques todos los scripts cuando hay una hoja de estilo que aún se está cargando y analizando. Webkit bloques los scripts sólo cuando intentan acceder a ciertas propiedades de estilos que pueden verse afectadas por hojas de estilo no cargadas (unloaded).

Rendizerizar la construcción del árbol

Nieñas se construye el árbol DOM, el navegador construye otro árbol, el árbol de renderizado. Es la representación visual del documento. Su propósito es permitir pintar los contenidos en su orden correcto.

Firefox llama "frames" a los elementos del árbol de renderizado. Webkit utiliza el término renderizador o objeto de renderizado.

Un renderizador sabe cómo diseñarse y pintarse a sí mismo y a sus hijos. La clase RenderObject de Webkit, la clase base de los renderizadores, tiene la siguiente definición:

```
class RenderObject{
    virtual void layout();
    virtual void paint(PaintInfo);
    virtual void rect repaintRect();
    Node* node; //the DOM node
    RenderStyle* style; // the computed style
    RenderLayer* containingLayer; //the containing z-index layer
}
```

Cada renderizador representa un área rectangular que generalmente corresponde al cuadro CSS de un nodo. Incluye información geométrica como ancho, alto y posición.

