Exercise1.1:

1.When a user enters an URL in the browser, how does the browser fetch the desired result ? Explain this with the below in mind and Demonstrate this by drawing a diagram for the same.(2-3hours)

a. What is the main functionality of the browser?

b. High Level Components of a browser.

c. Rendering engine and its use.

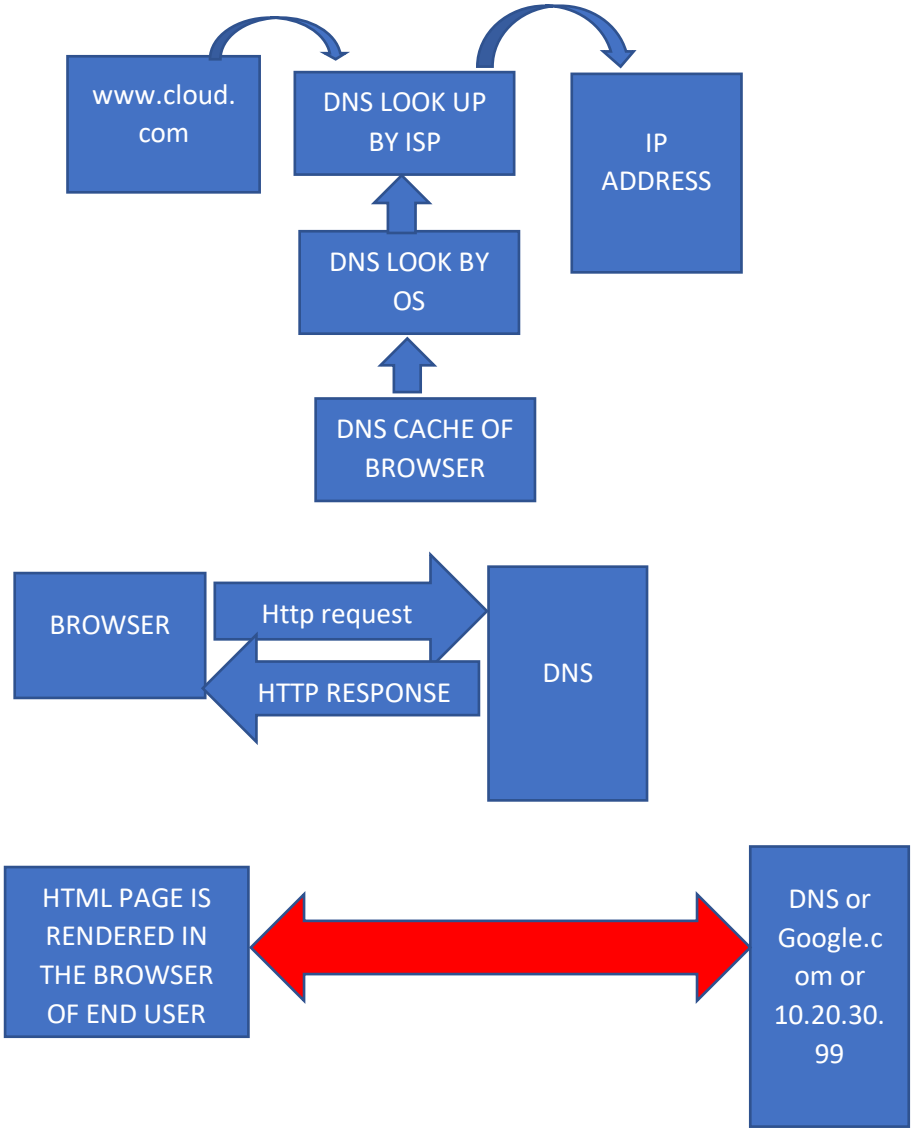
d. Parsers (HTML, CSS, etc)

e. Script Processors Tree constructing.

f. Order of script processing .Layout and Painting

==========

1.When a user enters an URL in the browser, how does the browser fetch the desired result.



Detailed steps for understanding

Step1: Enter a URL in WebBrowser

URL Prasing

- 1.When we search anything in the browser it checks whether it is url or search.
- 2.If it is URL then it checks for the Protocol (http(port-80) and https(port 443)).
- 3.if we didn't mention the Protocol then browser checks for preloaded HSTS(HTTP STRICT TRANSPORT SECURITY)list.
- 4.if the searched website is in that list, browser will use "Https"' otherwise "http".
- 5.If it "SEARCH" will check in cache.

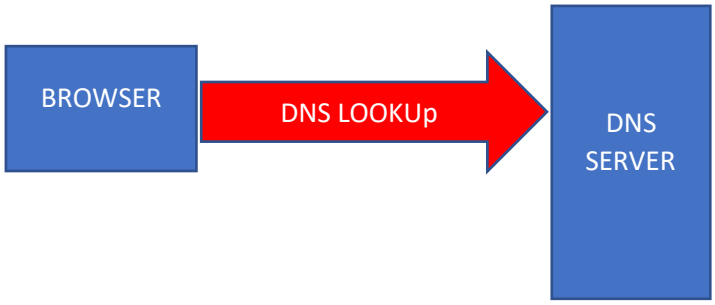
Step2:

Every website will be having unique IP Address,

Domain name linkedin.com and https :port 443 IP ?(finding for IP)

- 1.Browser checks if the domain is in its cache
- 2.if not found, the browser calls gethostbyname library function(varies by os)to do the lookup.
- 3.if not found ,then it makes a request to the DNS Server
- 4.if not found, then it makes a request to the ROOT DNS server and get the IP
- 5.Generic top level domains were created(.com/.org/.net etc)more than 13 root servers in the world

The browser(linkedin.com) lookup for the IP Adrees and DOMAIN NAME



Step3:

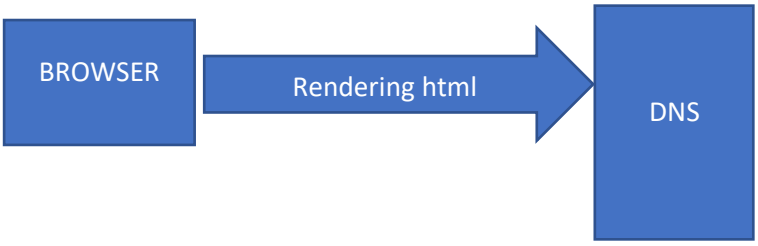
Browser sends http Request to the server

Step4:

Browser sends back the http response to the browser

Step5:

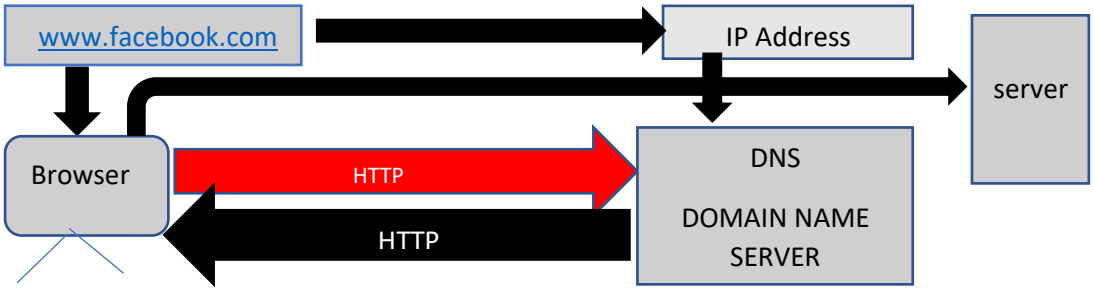
The Browser begins Rendering html



Step6:

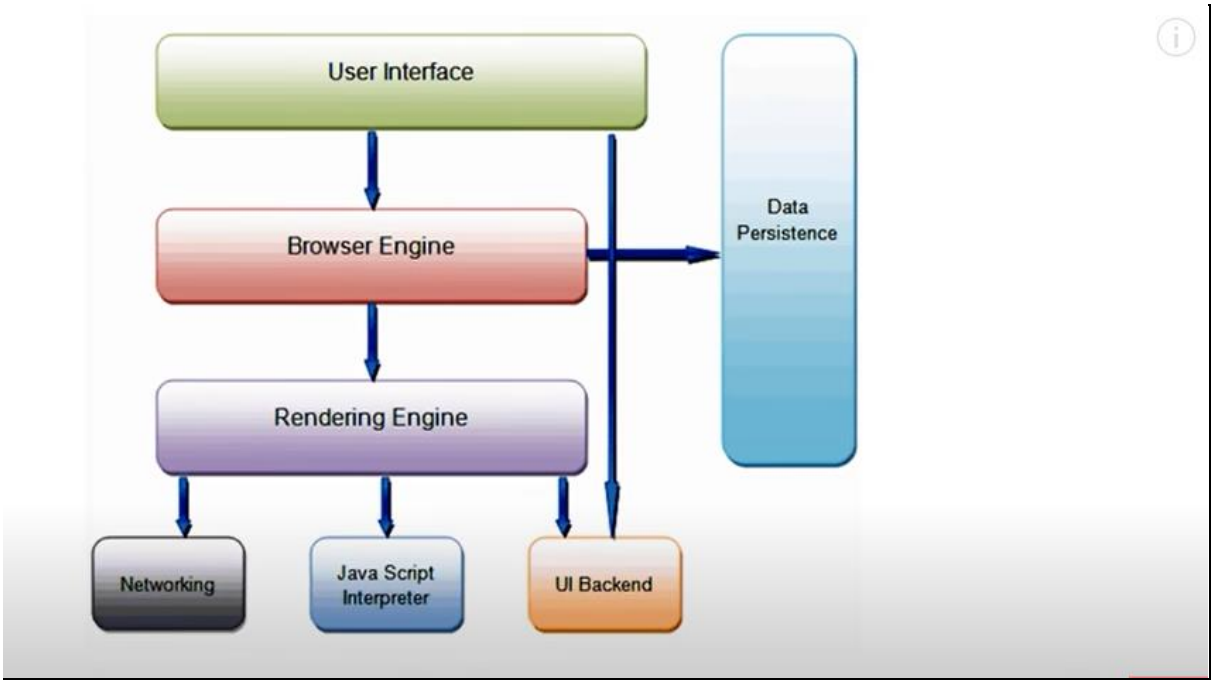
The browser sends request for additional objects and The browser shows the content of the url.

a)What is the main functionality of the browser?



- Browser will interact with the DNS as in the example www.facebook.com there Facebook.com Is the "DOMAIN NAME"
- In DNS or domain name we will be having the "IP ADDRESS"
- IP Adress is the webserver's address where requested webpages will be stored
- DNS server will be giving the address to browser
- Browser will send the address to web server then browser is connected with webserver
- Where facebook data is available then browser requested webpage fetches the data and gives to user

b. High Level Components of a browser



Browser's Components :

The browser's main components are :

1. The user interface:

This includes the address bar, back/forward button, bookmarking menu, etc. Every part of the browser display except the window where you see the requested page.

2. The browser engine: marshals actions between the UI and the rendering engine.

3. The rendering engine :

responsible for displaying requested content. For example if the requested content is HTML, the rendering engine parses HTML and CSS, and displays the parsed content on the screen.

4. Networking:

For network calls such as HTTP requests, using different implementations for different platform behind a platform-independent interface.

5. UI backend:

Used for drawing basic widgets like combo boxes and windows. This backend exposes a generic interface that is not platform specific. Underneath it uses operating system user interface methods.

6. JavaScript interpreter.

Used to parse and execute JavaScript code.

7. Data storage.

This is a persistence layer. The browser may need to save all sorts of data locally, such as cookies. Browsers also support storage mechanisms such as localStorage, IndexedDB, WebSQL and FileSystem.

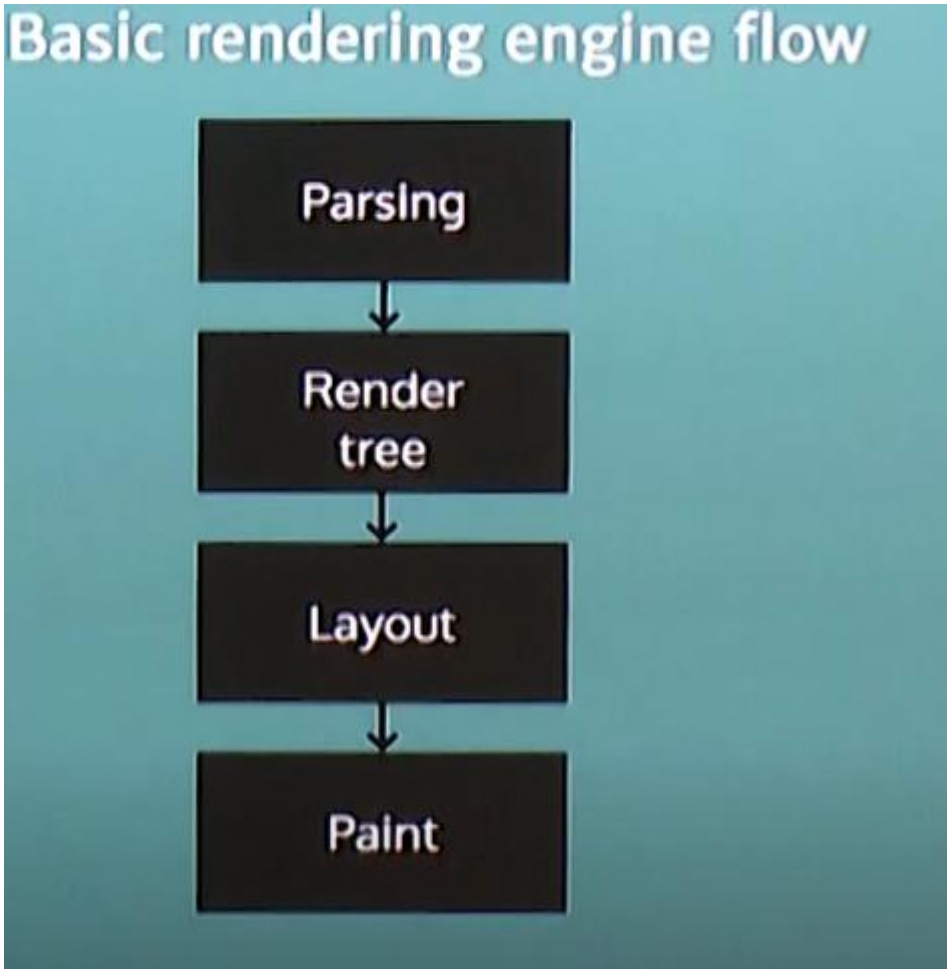
c.Rendering Engine And Its Use.

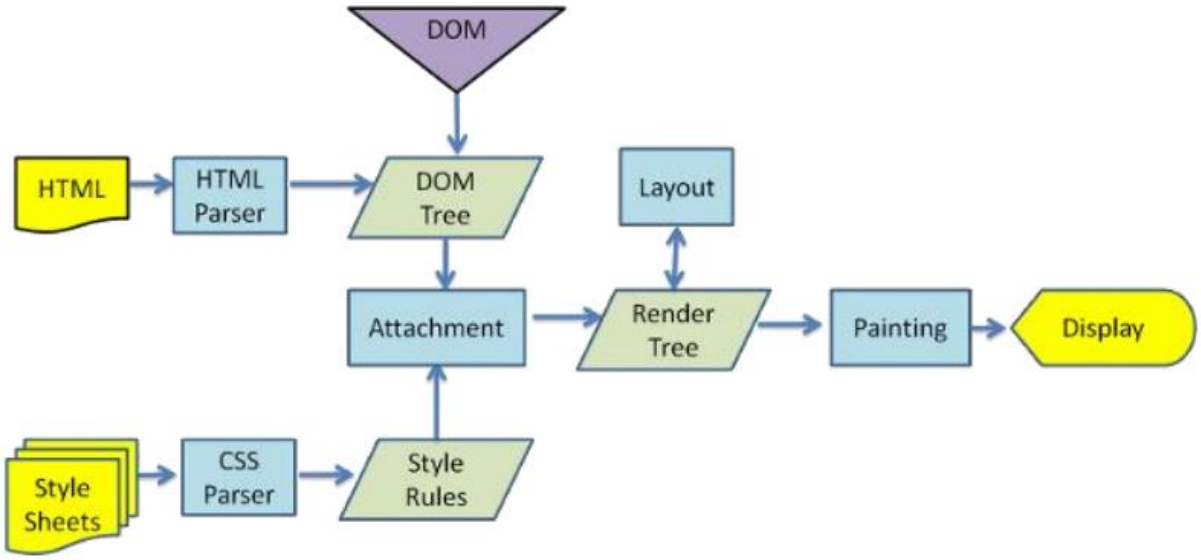
The rendering engine

The web browser that's responsible displaying content.

Most commonly, that involves parsing HTML (the language that describes the structure of web pages) and CSS (the language that describes how HTML should be styled), and then rendering the web page the HTML & CSS describes.

The responsibility of the rendering engine, rendering, that is display of the requested contents on the browser screen. By default the rendering engine can display HTML and XML documents and images.





Render tree construction

While the DOM tree is being constructed, the browser constructs another tree, the render tree. This tree is of visual elements in the order in which they will be displayed. It is the visual representation of the document. The purpose of this tree is to enable painting the contents in their correct order.

Firefox calls the elements in the render tree "frames". Webkit uses the term renderer or render object. A renderer knows how to layout and paint itself and it's children.

d.Parsers (HTML, CSS, etc)

HTML PRASER:

- The job of the HTML parser is to parse the HTML markup into a parse tree.
- HTML cannot be parsed easily by conventional parsers, since its grammar is not context free. HTML cannot be parsed by XML parsers.
- HTML cannot be parsed using the regular top down or bottom up parsers.
- Unable to use the regular parsing techniques, browsers create custom parsers for parsing HTML.

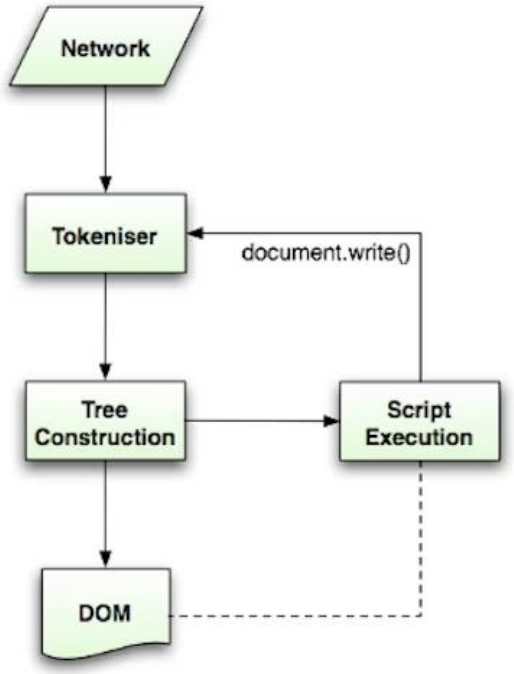
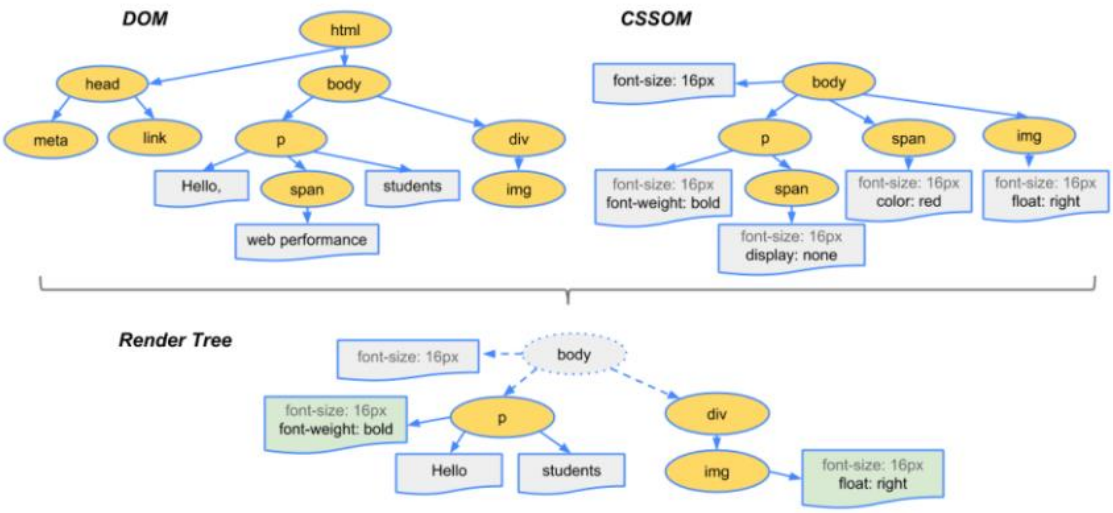


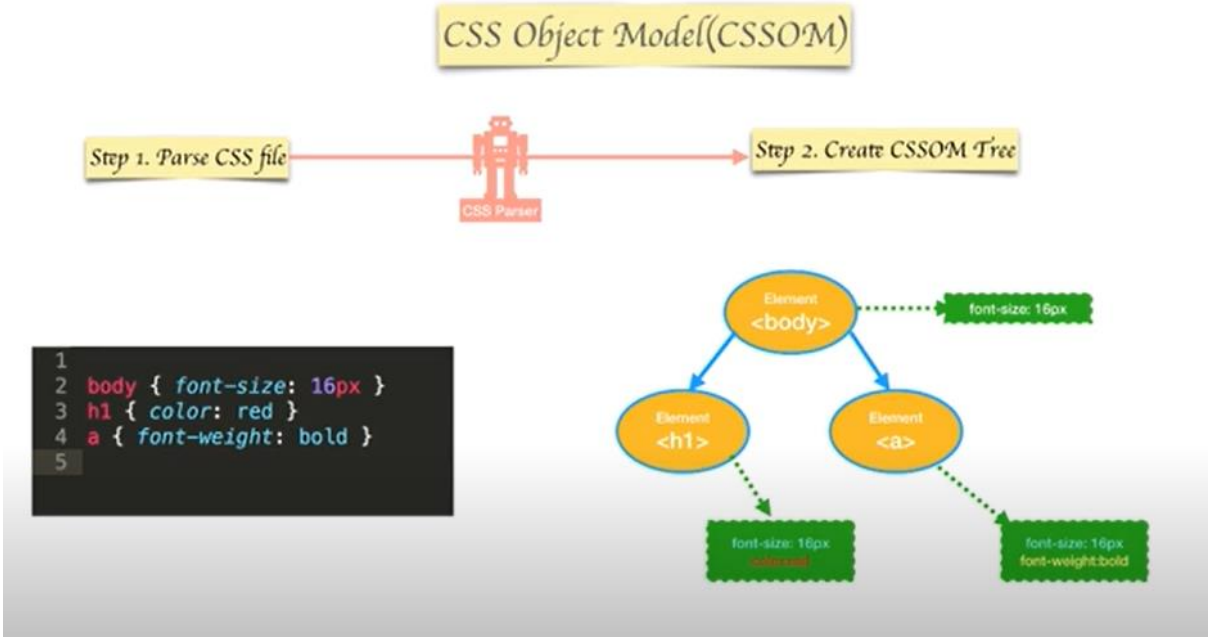
Figure : HTML parsing flow (taken from HTML5 spec)



Combining the CSSOM + DOM trees to make the render tree

CSS Parser

- The data structure that the browser turns the CSS into is called the CSSOM.
- CSS rules are read from right to left, meaning that if we have something like: section p { color: blue; }, the browser will first look for all p tags on the page and then it will look if any of those p tags have a section tag as a parent. If that's the case, it will apply the CSS rule



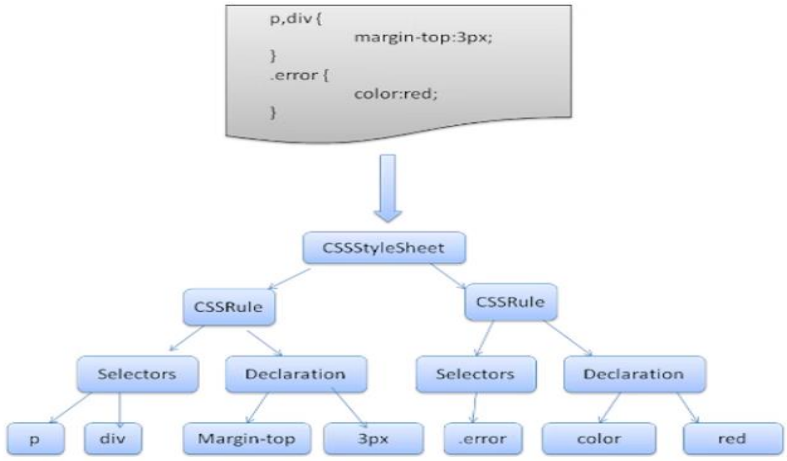


Figure : parsing CSS

e. Script Processors.Tree constructiong.

Script Processor

The model of the web is synchronous. Authors expect scripts to be parsed and executed immediately when the parser reaches a <script> tag. The parsing of the document halts until the script has been executed. If the script is external then the resource must first be fetched from the network - this is also done synchronously, and parsing halts until the resource is fetched. It is specified in HTML4 and 5 specifications. Authors can add the "defer" attribute to a script, in which case it will not halt document parsing and will execute after the document is parsed.

HTML5 adds an option to mark the script as asynchronous so it will be parsed and executed by a different thread.

Tree construction

The CSSOM and DOM trees are combined into a render tree, which is then used to compute the layout of each visible element and serves as an input to the paint process that renders the pixels to screen. Optimizing each of these steps is critical to achieving optimal rendering performance.

on constructing the object model, built the DOM and the CSSOM trees based on the HTML and CSS input. However, both of these are independent objects that capture different aspects of the document: one describes the content, and the other describes the style rules that need to be applied to the document.

The DOM and CSSOM trees are combined to form the render tree. Render tree contains only the nodes required to render the page. Layout computes the exact position and size of each object. The last step is paint, which takes in the final render tree and renders the pixels to the screen.

f. Order of script processing .Layout and Painting

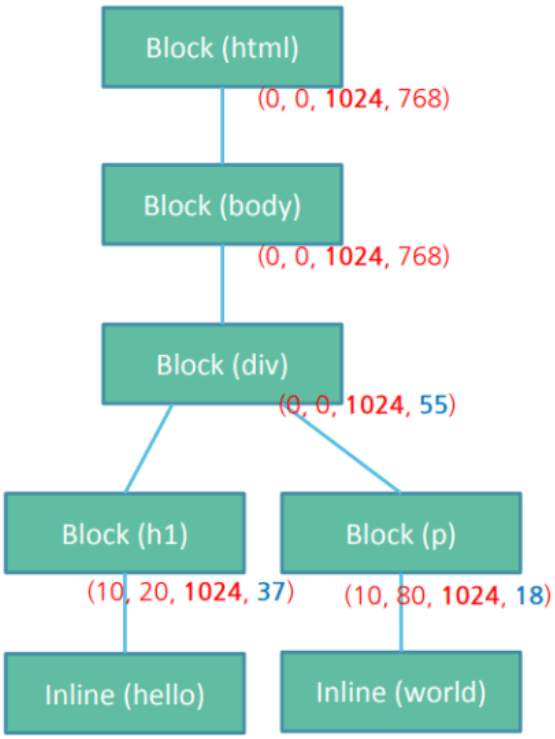
The order of processing scripts and style sheets

1.Scripts

2.Speculative parsing

3.Style sheets

Layout:
The layout (also called reflow) peace will be in charge to calculate the positions and dimensions of each node on the screen.



Painting
which converts each node in the render tree to actual pixels on the screen. This step is often referred to as “painting”, “rasterizing.” or “repainting”. It will repaint itself.

- Painting order is:
- background color
- background image
- border
- children
- outline