**HOW DOES THE BROWSER ACTUALLY RENDER A WEBSITE?**

Some of components that make up browser for rendering are:

1.Bindings - api - mac or windows

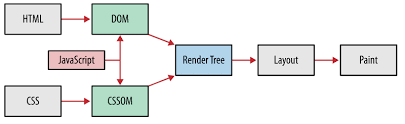
2.Rendering - parsing,layout,painting

3.Platform - windows or osx

4.Javascript virtual machine

**High level flow**

1.Parsing:



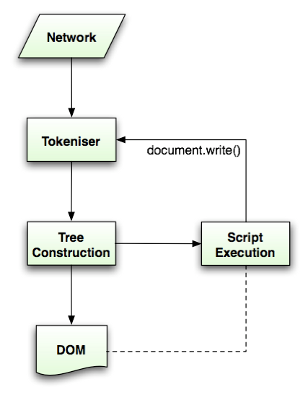
- html is forgiving by nature

- parsing isnt straight forward

- can be halted

- will do speculative parsingits reentrant.

Parsing flow:



* <script> ,<link> , <style> - halt the parser as script can alter the document
* speculative parsing -> external images, scripts,css
* <script/> at the bottom -> parse uninterrupted ,faster to render - defer and async attributes - trade off

1.Parsing html document

2.Render / frame tree (dom +cssom)

- Multiple trees - renderobjects , renderstyles, renderlayers, lineboxes

- In render tree - not in dom - tags like p,string,em

- dom node to render object - > visual output , geometric info, can layout &paint,holds style and computed metrics

**Calculating visual prop**:

- combine all styles

- defaults ,external,inline

- comlexity around matching rules for each elemnt

- style computation

3.Layout:

- recursive process

- batch layout

- immediate layout

**Performance insights 1**

->Take note frm browser and batch -> act like the browser and batch your dom chances , do all your reads in one pass, followed by writes.

Real world - fast dom , preventing layout thrashing & most modern js frameworks do this internally

4.Paint

- will take layed out render tree

- created layers

- incremental process

- builds up over 12 phases

--> painting - bit,map from each layer - composites the texture into final image to render to the screen

**Performance insights 2**

- speeds up first paint times

- external js and css can block

- delta last bitmap