**How does the browser render a website**

**Browser**

One of the complex applications we use.

**Binding**

Lot of operating system stuff and the networking and APi involves in binding operation.

**Rendering:**

Parse HTML + Parse CSS -🡪Render Tree -🡪 Layout -🡪Paint

**Parsing HTML:**

HTML is forgiving by nature

Parsing is not straight forward

Parsing can be halted

Will do speculative parsing

It’s reentrant.

**Parsing Flow;**

Script Execution

Tokeniser -🡪 Tree Construction -🡪DOM

**Parse Tree :**

**Parsing tree to DOM tree occur in this process.**

It is representational of HTML . It is one to one.

**DOM Tree**

Document Object Module Tree. Processing the HTML markup and building the DOM tree. HTML parsing involves tokenization and tree construction. HTML tokens include start and end tags, as well as attribute names and values.

**DOM tree to Render Tree process occur. So totally 4 trees**

**Speculative Parsing**

Parsing external images , scripts and CSS

<script src=”script.js”>

<img src=’bird.gif’/>

<link href=”style.css

**Reentrant**

It means the parsing process can be interrupted.

**DOM + CSSOM**

Combines the two object models , style resolution

**Bulding the CSSOM**

The critical rendering path is processing CSS and building the CSSOM tree. The CSS object model is similar to the DOM. The DOM and CSSOM are both trees. They are independent data structures. The browser converts the CSS rules into a map of styles it can understand and work with. The browser goes through each rule set in the CSS, creating a tree of nodes with parent, child, and sibling relationships based on the CSS selectors.

Multiple tress Render Objects , render styles , render layers and line boxes.

Node display can be hidden by none property.

**DOM Node to Render Object**

It is visual ouput

Geometric Info, It Can layout and paint, Holds style and computed metrics.

Rendering steps include style, layout, paint and, in some cases, compositing. The CSSOM and DOM trees created in the parsing step are combined into a render tree which is then used to compute the layout of every visible element, which is then painted to the screen.

**Calculating visual Properties**

The browser calculate visual properties it combine all styles , so that the browser default styles the external styels you are linking to , any inline style elements or styles take it to account.

**Layout**

**Layout computes where a node will be on screen**

It is recursive Process means it will work its way. Based on the position and size Layout its children

Will batch Layout- incremental Layouts. The browser will intelligently batch changes.

Immediate Layout- Doing font size change will relay out the entire document. Same with browser resize.

FastDom, preventing layout from thrashing. Most modern JS frameworks do this internally.

**Painting**

**Painting computes bitmaps and composites to screen.**

This is what takes all the information from the render tree and will actually do the calls to paint something and give the visual output.

It creates layers from render objects and Position nodes, transparency, overflow, canvas, videos. builds up over 12 phases.

Produces a bitmap from each layer. Bitmap is uploaded to the GPU as a texture. Composite the texture into final image to render to the screen.

Inline critical CSS

The most important bits of your site/app.

External js and css can block.