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)

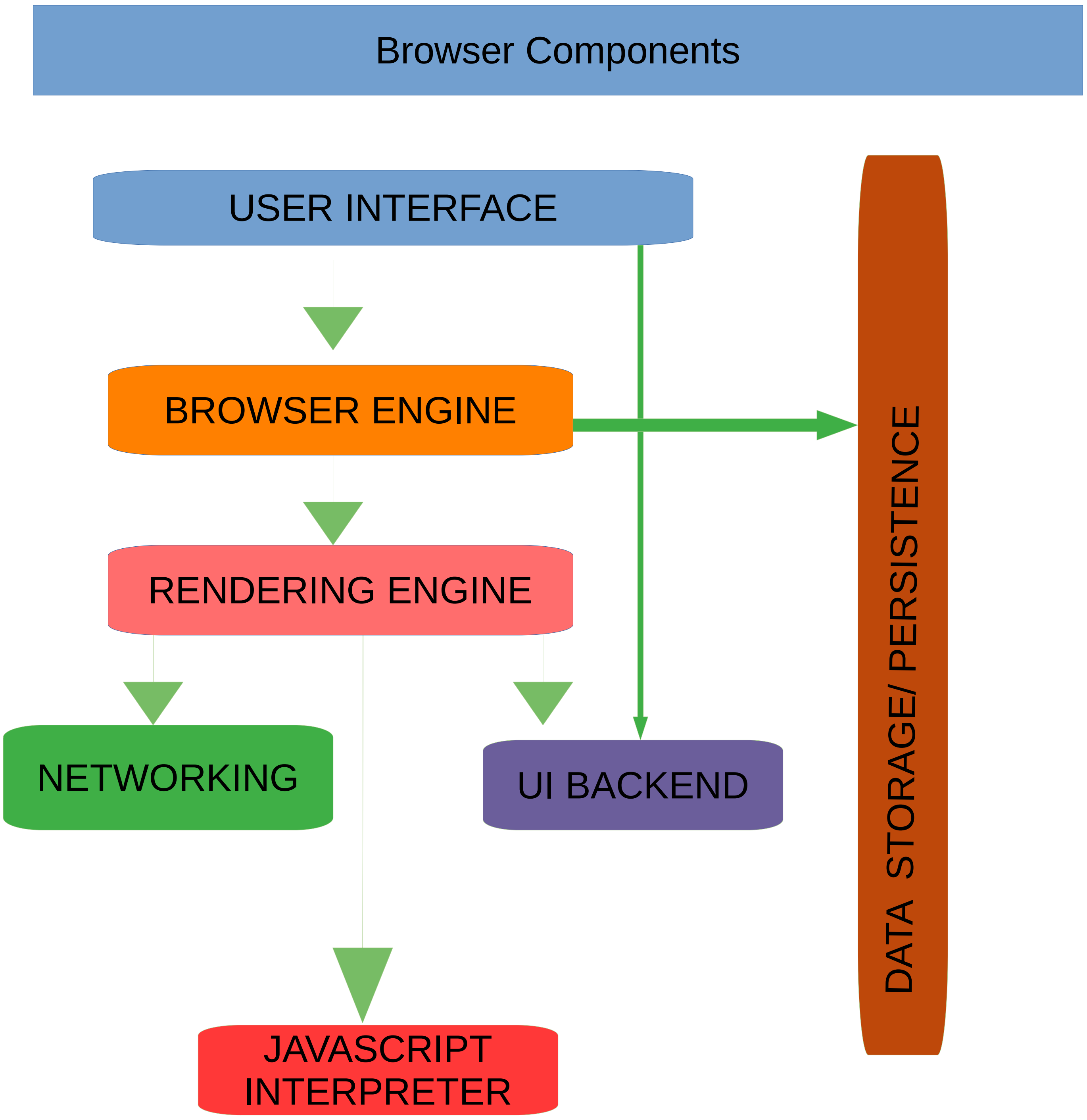
a. What is the main functionality of the browser?
b. High Level Components of a browser.
c. Rendering engine and its use.
d. Parsers (HTML, CSS, etc)
e. Script Processors
f. Tree construction
g. Order of script processing
h. Layout and Painting Guidelines

1.Submit this assignment on GIT - Answer should be in readme File (with images) on GIT.
2.Candidates should be able to explain how a browser works.
3.What are the high level components of a browser?
4.How each component works with each other. (For example: Networking component is the one which makes HTTP calls, Data storage component is a browser's persistence layer which saves data locally such as Cookies and Local Storage.
5.How Parsing works and its importance.
6.The order of execution of scripts.

Outcome:

1.Under the hood understanding of how a browser works.
2.What are the features a browser provides?
3.What a browser is capable of doing.
4.How a web page is translated from a string in a URL to a webpage

Answer to question b. High level components of browser



Browser components:
1. User interface : This includes the address bar, back forward buttons, refresh buttons, bookmarkings menus, settings menu etc.
2. The Browser engine: Heart of browser, manages action between User interface and rendering engine and manages data storage and persistence. Memory related interface with OS.
3. The rendering engine: Responsible for displaying the requested URL content. Rendering engine communicates with network, has Javascript interpreter and works with UI backend
4. Networking : For network HTTP requests such as get, put, post etc.
5. UI Backend: used for drawing basic widgets like combo boxes and widgets. For implementation of capture, target and bubble phases(?)
6.JS interpreter: Used to parse and execute JS code.
7. Data storage: This is a persistence layer to store data locally, such as cookies.

Browsers run multiple instance of rendering engines- one for each tab.

Answer to question a. The main functionality of browser is:
1. To provide a simple user interface to input, save and navigate by means of URLs.
2. Provide means to set preference for data management and privacy settings.
3. Request for webpages by converting URL string to specific unicode HTTP request (?)
4. Render the received packages (HTML,XML, PDF etc.)
5. Allocate memory and its management.
6. In case of HTML, parse it, parallely request for CSS, Images, JS and other linked packages.
7. Start drawing and presenting the page.

Answer to question C:

Rendering Engine:

To render the display of the requested contents on the browser. In itself the rendering engine can display HTML, XML, Images, PDF or other file type if supported plugins are installed.
Different browser may use different rendering engines, however since most of them comply with W3C specification, the rendering of HTML does not vary much from browser to browser.
Example: Internet explorer : Trident
Firefox : Gecko
Chrome: Blink(fork of Webkit)
EDGE:

How it works :
The rendering engine will start getting the content of the requested document from the networking layer. After this the basic flow of the rendering engine is

1. Parse HTML to construct DOM tree.
2. Render Tree construction
3. Layout the render tree
4. Painting the render tree

Rendering engine will parse the HTML and convert its elements into DOM nodes making a content tree. The engine will parse the style data, both in external css and inline style elements. Content tree combined with styling instructions will be used to generate the render tree.

Render tree will contain rectangles with visual attributes like color, text size and dimensions.

After the construction of render tree layout process starts. Which gives each node the exact coordinates where it should

It should appear of screen.

Next process is painting – The render tree is will be traversed and each node will be painted using the UI backend layer.

The rendering engine starts will try to display the content as soon as possible. It will not wait until all HTML is parsed before starting to build and layout the render tree.

Answer to question D,E:
Parsing means translating a document into a structure that code can use. The result of parsing usually a tree of nodes that represent structure of the document called the syntax tree or parse tree.
Parsing can be subdivided into lexical analysis and syntax analysis.
Lexical analysis is process of breaking the input into tokens or valid building blocks. Syntax analysis is the applying of language syntax rules. The lexer/tokenizer breaks input into valid tokens and the syntax parser is constructs the parse tree based on document structure and language syntax rules.
Parsing is iterative. The parser will ask for a new token and match it with the syntax rules. If the rule is matched, a node corresponding to token is added to the parse tree and the parser will ask for next token.
In case no rule matches, token is store internally and parser ask for next token until a rule matching all the internally stored tokens is found. In case no rule are found the parser with raise exception.

Job of HTML parser is the HTML markup into a parse tree.
Vocabulary and syntax of HTML are defined in specifications created by W3C orgainsation. :
https://html.spec.whatwg.org/multipage/
https://html.spec.whatwg.org/multipage/syntax.html

The output of HTML parser is tree of DOM element and attribute nodes. Document Object Model.

CSS parsing:
CSS parsing results in Stylesheet object tree. Each object contains CSS rules. The CSS rule objects contain selector and declaration objects and other objects corresponding to CSS grammar.
Answer to Question G.

The order of processing scripts and style sheets
Scripts – Scripts are parsed and executed immediately when parser reaches <script> tag. The parsing halts until the script is executed. If the script is external the resource must first be fetched from network, parsing halts until resource are fetched. Unless defer attribute is added to script.

While the parsing is halted, waiting for external resources another thread parses rest of the documents for more resources and starts loading them. This is called speculative parsing.

Style sheets processing vary between browsers. Although style sheet don't change structure of DOM tree, they script attributes can be calculated incorrectly if style sheets are not available. Some block all scripts when there is a style sheet still being loaded and parsed.

The next step is Render tree construction which is constructed using DOM tree and applying Style tree rules over the nodes.

The Rendering tree becomes input for layout process.

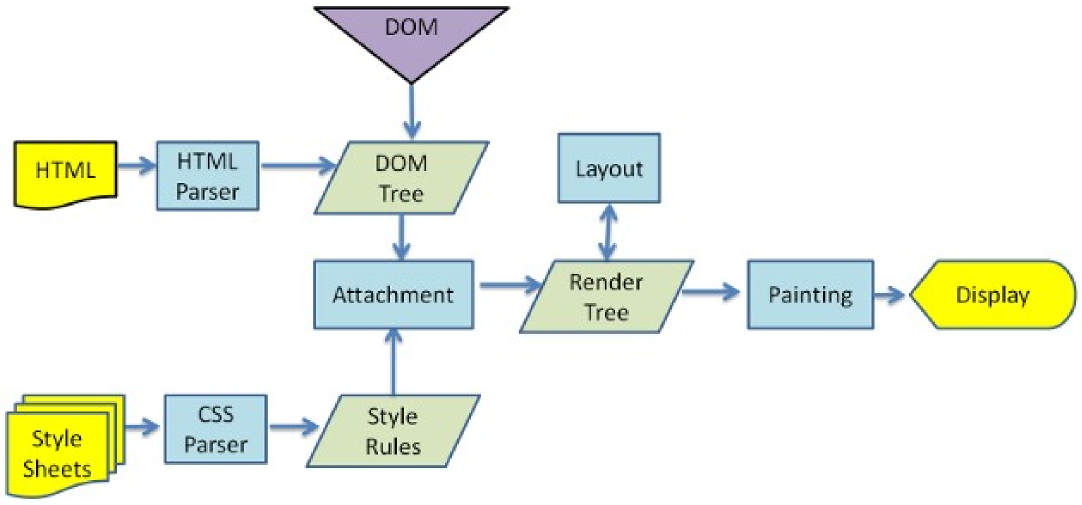
Answer to question H.

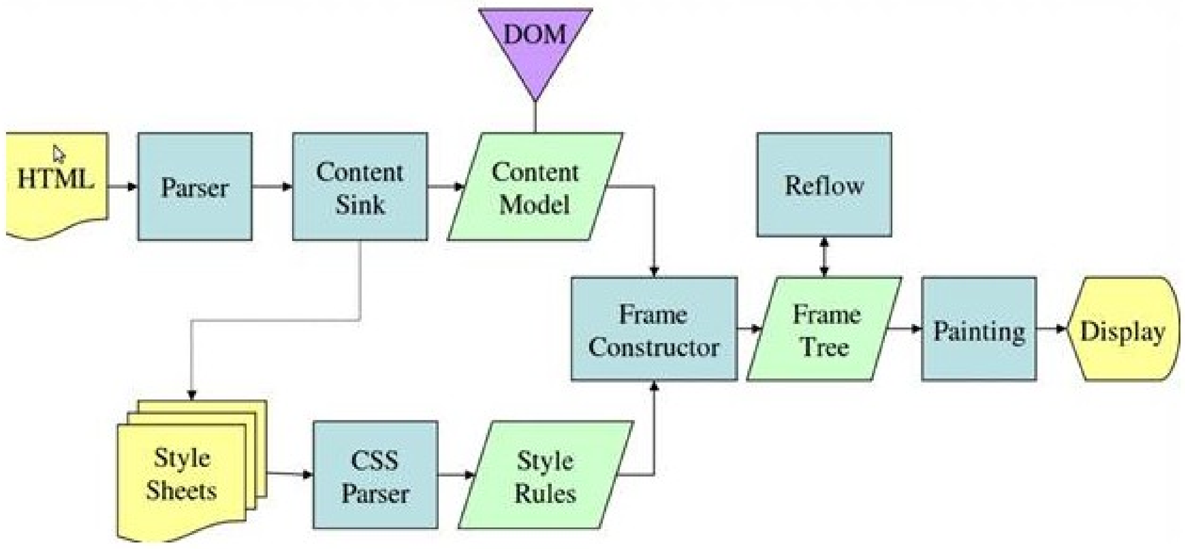
The layout or reflow is a recursive process. It begins at the root renderer, which is <html> element. Layout continues through some or all of frame hierarchy, computing geometric information for each renderer that requires it. Position of root renderer is 0,0, that is top left of viewport. Its dimensions are of the viewport i.e visible part of window.

The layout process follows following pattern:
1. Parent renderer determines its own width.
2. Parent goes over children and:
    a. Place the child renderer -set its x & y
    b. Calls child layout if needed to calculate child's height.
3. Parent uses children's accumulative heights of margins and padding to set its own height – this will be used by the parent renderer;s parent.
4. Sets its dirty bit to false.

Painting stage the render tree is traversed and the renderer's paint() method is called to display content. Paints uses the UI infrastructure component.

Following are the main flow examples from Webkit and Gecho rendering engine


DOM
HTML
HTML Parser
DOM Tree
Layout
Attachment
Render Tree
Painting
Display
Style Sheets
CSS Parser
Style Rules


DOM
HTML
Parser
Content Sink
Content Model
Reflow
Frame Constructor
Frame Tree
Painting
Display
Style Sheets
CSS Parser
Style Rules