July 20, 2020

Team:

Abdullah | Qazi Arsalan | Mahnoor | Sabir

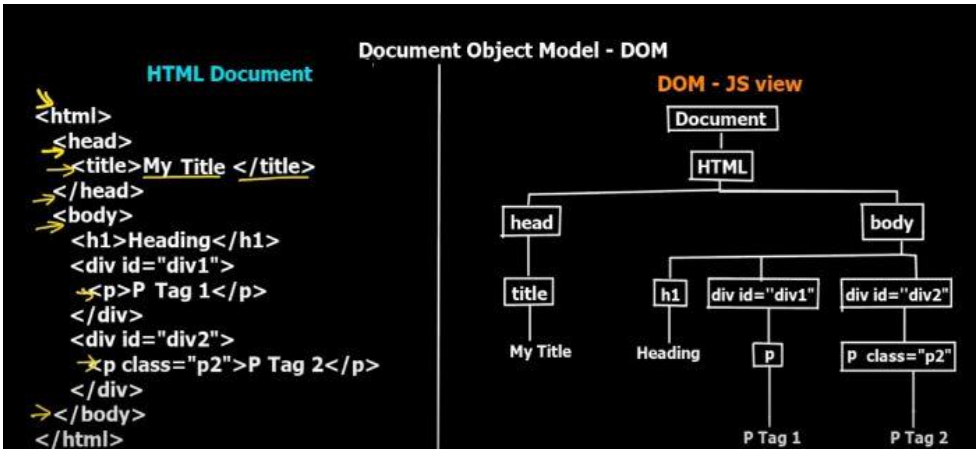
Findings from the Article

Trees and tree traversing

There is a Concept of DOM (document object model) in many well-known Applications. Elements in the DOM are organized into a tree data structure that can be traversed to locate or modify elements.

- UI Frameworks:

  In web development, we have an html page having tags in it. It is static only and if we want to perform some operation like button click so we use some programming language like JavaScript. We saw that JavaScript reads the html code in the form of tree having tags as vertices and leaves. You could implement BFS or DFS to traverse through all nodes. React maintains a virtual DOM, which means that actual DOM will update after the virtual DOM has completed comparing its pre and post snapshot. In short, virtual DOM take snap of it before changing any element and after then, actual DOM is dealed accordingly. It's all done by a "Diffing Algorithm".



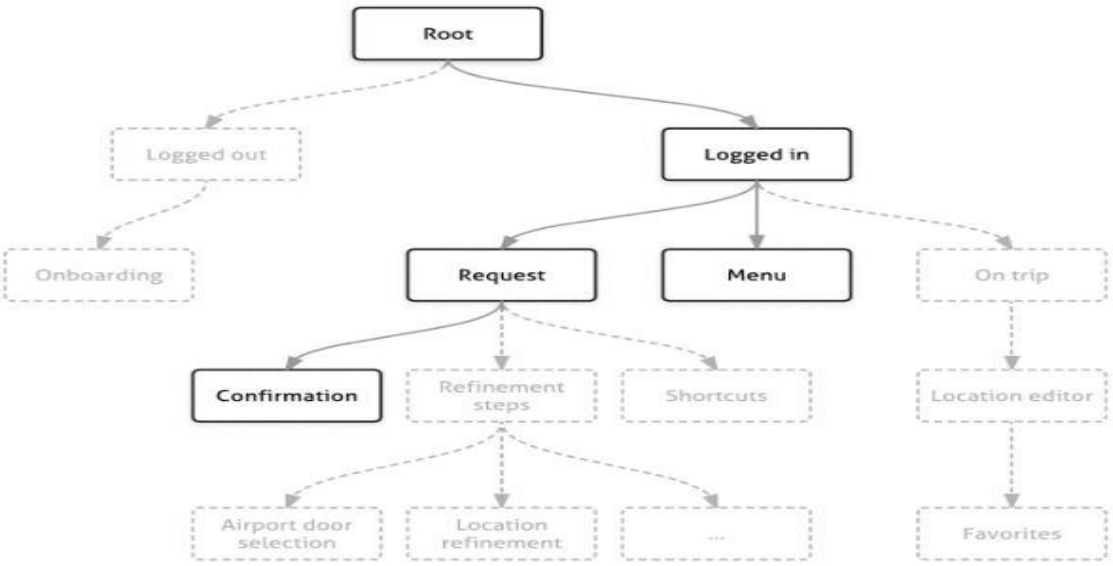
- UBER:

  UBER Mobile Architecture, RIBs (Router , Interactor and Builder) also uses the same concept of tree traversal

  Router: Listens to Interactor and outputs into attaching or detaching child RIB's

  Builder: Instantiate Builders of RIB's children

  Component: They assist Builders by providing external dependencies. Components are injected to child's RIB to give child access to parent dependencies

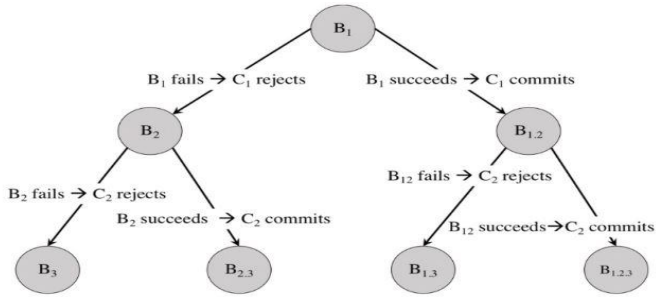


Similar concept, as user make modification in the mobile app like add route, similarly, nodes get attached and if he left a route, node detaches.

- Decision Trees:

  In Mobile Applications, we had to display several different screens. The rules were unusually complex due to series of compliance checks and user choices that we need to take in account.

  They were tired of if else statements so they used decision trees. They needed to build an implementation for edges to execute. Here's how the decision tree looked like - the edges being results of rules executed (either binary outcome or value ranges) and the leaf nodes marking what screen to transition to.



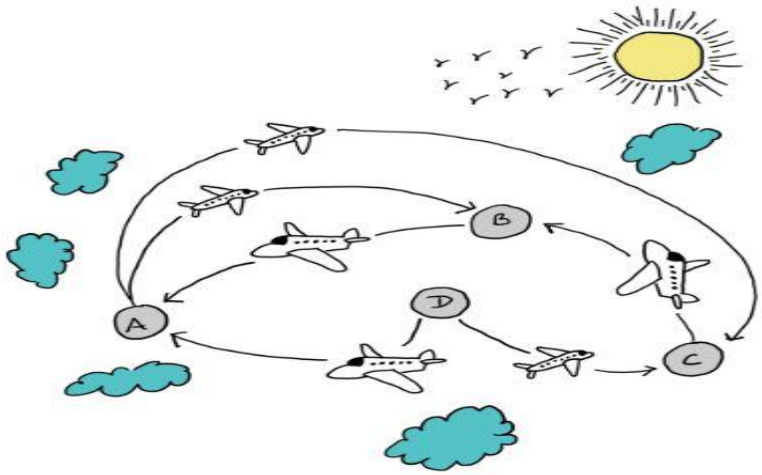
  Uber's mobile build system, called SubmitQueue, also utilizes decision trees built on the fly. The Developer Experience team took an innovative approach by predicting merge conflicts and queued builds accordingly, using speculation graphs.

Weighed graphs and shortest paths

Graph which have weight on their edges and shortest path means, you can find shortest path from any edge.

- Skyscanner:

  Skyscanner is a china's largest travel agency owned by Trip.com. When a person want to visit multi cities so their multi-city algorithm comes in handy. Multi-city deals are calculated by using shortest path algorithms like Dijkstra or A*. Flight routes are represented as a directed graph, with each edge having a weight of the cost of the ticket. Calculating the cheapest price option between two cities was done via an implementation of their own modified A* Algorithm.



Sorting:

- Skype:

  From the article, we came to know when skype connected to the network, contacts arrives in a burst so there is a need to sort them so according to the author, they used insertion sort to sort them on the basis of names.

  Insertion sort can be useful when streaming real-time data in large chunks and building real-time visualization for these data sources.

  Merge sort can work well with divide-and-conquer approaches if it comes to large amounts of data stored on different nodes.

  Sorting is used by us daily like our phone sorts the data in its memory as it arrives.

Hash tables and hashing:

- Distributed Systems:

  Distributed systems has to store a lot of data as compared to a single pc or node. So technique we use is resharding.
  Sharding is simply distribution of data on multiple databases rather than single base.
  Many distributed system have data replicated over its nodes, so for this we use voting technique where a certain number of nodes needs to get the same result, for the operation to be successful. This is called the quorum.

Interview:

Author of the article says that

- You should not need to memorize the algorithms
- You should know what an algorithm is, and should be able to come up with simple ones on your own, like a greedy one.
- You should know the basic data structure only that are pretty common.
- Do not memorize the DFS or BFS or Exotic Algorithms like RED-Black trees.
- If you are working in any tech company, you will came across all data structures when need in a code base. You will often find yourself reaching to the right data structure.

“Data structures and algorithms are a tool that you should use with confidence when building software”