REACT JSX + DOM

APRIL 2022

YONATAN BENEZRA

A React Hello World

Now for the fun part - writing the code! Open up your App.js file. This will be our top-level, or root, component for our application.
Inside that file we already have some code written. We will need to import the following once again:

- React (which we need to do in every file that we use React in)
- The React logo
- The CSS file specific to this App component.
  We also have a function, App, that returns a bunch of what looks like HTML -- this is actually JSX!
  Finally, we will export the component so we can import it into other files, in this case, our index.js that was generated for us.

Let's go ahead and remove the logo import and change the JSX code so that it only returns the following:

```
<h1>Hello, World</h1>
import React from 'react'
- import logo from './logo.SVG
import './App.css'

const App = () => {
  return (
+ <h1>Hello world!</h1>
  )
}

export default App
```

JSX is an extension of JavaScript that allows you to write what looks like HTML directly in your JavaScript code.

You can't natively use JSX in the browser, but we will use a library called Babel to transpile (or convert) our JSX into regular JavaScript so that the browser can understand it (this happens under the hood).
JSX is optional in React, but you'll see it used in the vast majority of cases.

Okay, now you've written your first React code, but how do you see the output? Go back to your CLI and run the following:

```
npm run start
```

A webpage will pop up that displays your React web application.
It will not reload, meaning every time you change your code and save those changes, your application will refresh the changes automatically.

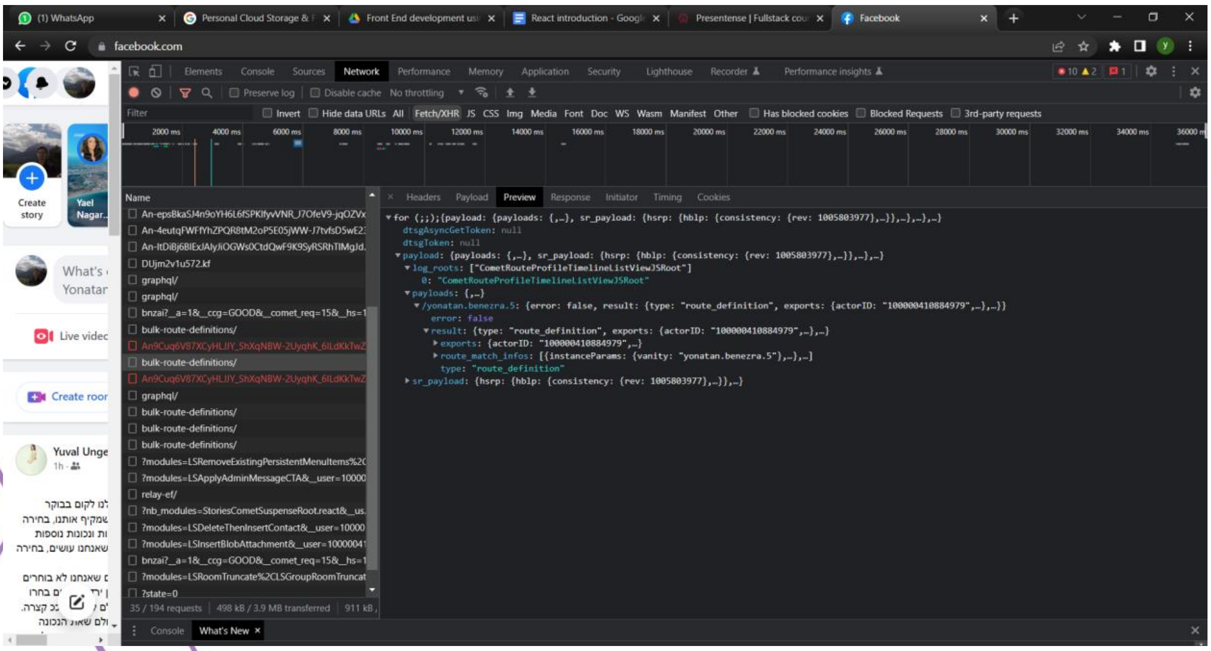
To exit out of the server that is running your app, press Ctrl+C.
It may be helpful to have two terminal windows or tabs open while you're developing React apps, because you can't write additional commands in the session where the server is running.

Dom & Virtual DOM

Let's talk about DOM and the virtual DOM, this will help us understand one of the reasons to use REACT.

There are **two things** which you must have a basic understanding of before reading this lesson:

- **What is an API?**
  API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other.
  Each time you use an app like Facebook, send an instant message or check the weather on your phone, you are using an API.



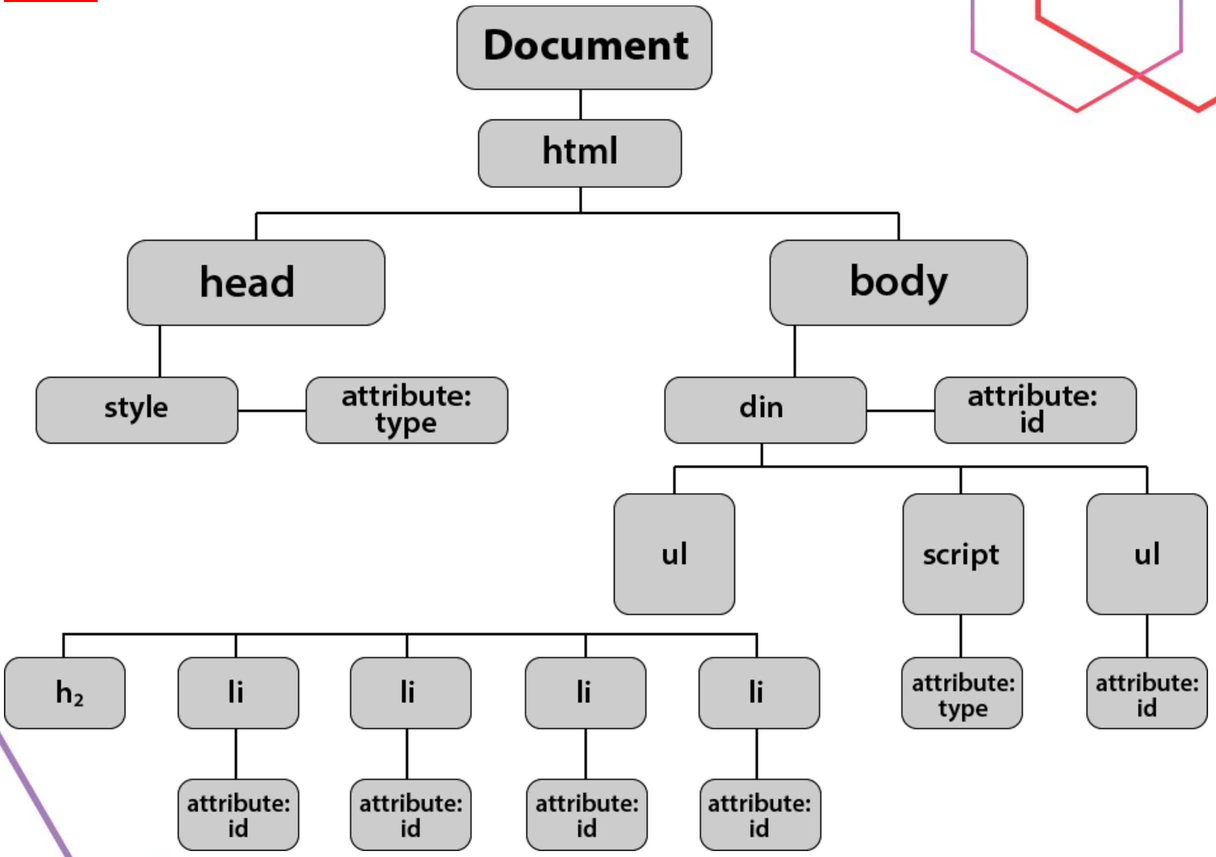
```
Name
An-epsBkaSJ4n9oYH6L6fSPKIfyVNR_J7OfeV9-jqOZVx
An-4eutqFWFfYhZPQR8tM2oP5E05jWW-J7tvfsD5wE2
An-ItDiBj6BIExJAlyJiOGWs0CtdQwF9K9SyRSRhTIMgJd.
DUjm2v1u572.kf
graphql/
graphql/
bnzai?__a=1&__ccg=GOOD&__comet_req=15&__hs=1
bulk-route-definitions/
An9Cuq6V87XCyHLJJY_ShXqNBW-2UyqhK_6ILdKkTwZ
bulk-route-definitions/
An9Cuq6V87XCyHLJJY_ShXqNBW-2UyqhK_6ILdKkTwZ
graphql/
bulk-route-definitions/
bulk-route-definitions/
bulk-route-definitions/
?modules=LSRemoveExistingPersistentMenuItems%2C
?modules=LSApplyAdminMessageCTA&__user=10000
relay-ef/
?nb_modules=StoriesCometSuspenseRoot.react&__us
?modules=LSDeleteThenInsertContact&__user=10000
?modules=LSInsertBlobAttachment&__user=1000004
bnzai?__a=1&__ccg=GOOD&__comet_req=15&__hs=1
?modules=LSRoomTruncate%2CLSGroupRoomTruncat
?state=0
35 / 194 requests | 498 kB / 3.9 MB transferred | 911 kB
```

```
▼ for (;;);{payload: {payloads: {,…}, sr_payload: {hsrp: {hblp: {consistency: {rev: 1005803977},…}},…},…},…}
    dtsgAsyncGetToken: null
    dtsgToken: null
  ▼ payload: {payloads: {,…}, sr_payload: {hsrp: {hblp: {consistency: {rev: 1005803977},…}},…},…}
    ▼ log_roots: ["CometRouteProfileTimelineListViewJSRoot"]
        0: "CometRouteProfileTimelineListViewJSRoot"
    ▼ payloads: {,…}
      ▼ /yonatan.benezra.5: {error: false, result: {type: "route_definition", exports: {actorID: "100000410884979",…},…}}
          error: false
        ▼ result: {type: "route_definition", exports: {actorID: "100000410884979",…},…}
          ▶ exports: {actorID: "100000410884979",…}
          ▶ route_match_infos: [{instanceParams: {vanity: "yonatan.benezra.5"},…},…]
            type: "route_definition"
    ▶ sr_payload: {hsrp: {hblp: {consistency: {rev: 1005803977},…}},…}
```

You can see in the above image that Facebook sent a request to its API to get certain data. My name is shown there!

- **What is a node?**
  A node is a basic unit of a data structure, such as a linked list or tree data structure.
  Nodes contain data, and may also contain links to other nodes.

DOM



DOM stands for Document Object Model and is an abstraction of a structured text.
For web developers, this text is an HTML code, and the DOM is simply called HTML DOM.

Elements of HTML become nodes in the DOM.
So, while HTML is a text, the DOM is an in-memory representation of this text.

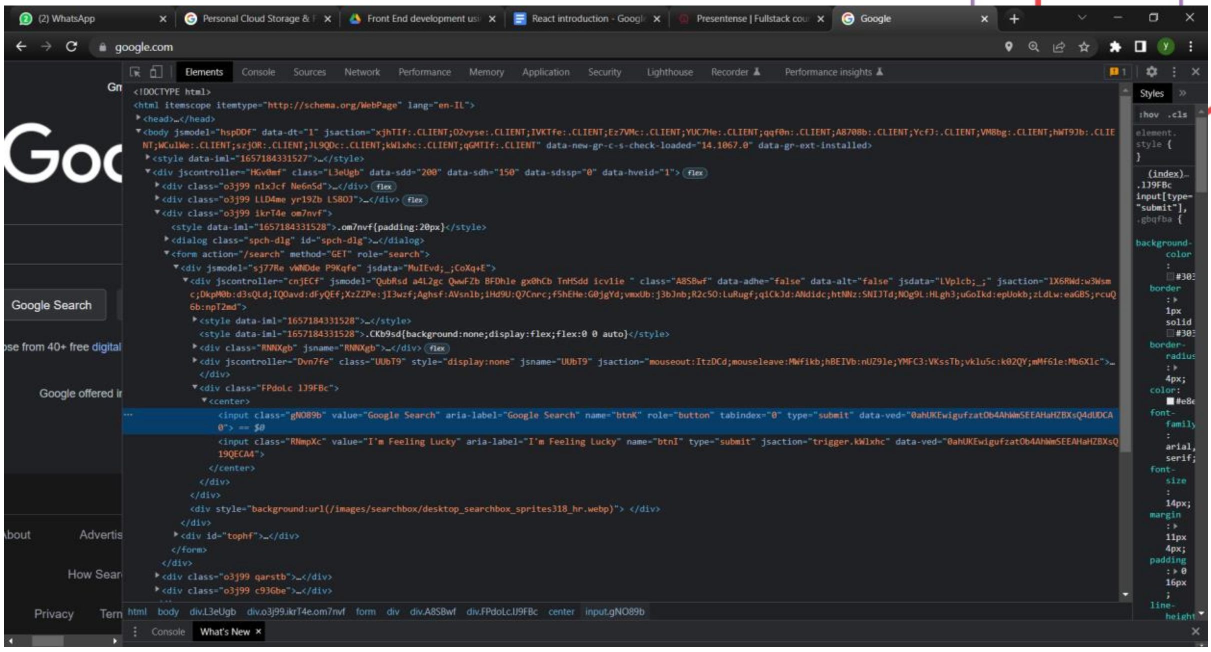
This can be compared to the way a process is an instance of a program.
You can have multiple processes of the same program, just like you can have multiple DOMs of the same HTML (e.g. the same page loaded on many tabs).

The HTML DOM provides an interface (API) to traverse and modify the nodes.
It contains methods like getElementById or removeChild. We will usually use JavaScript language to work with the DOM.

So, whenever you want to dynamically change the content of the web page, you will want to modify the DOM.

Note:
DOM is a specification of objects in browsers, widely agreed upon by all the major web browsers.

The DOM uses a tree structure, which means that to change a node in the middle of the structure, we need to change all the nodes above it as well.



```
<!DOCTYPE html>
<html itemscope itemtype="http://schema.org/WebPage" lang="en-IL">
 ▸<head>…</head>
 ▾<body jsmodel="hspDDf" data-dt="1" jsaction="xjhTIf:.CLIENT;O2vyse:.CLIENT;IVKTfe:.CLIENT;Ez7VMc:.CLIENT;YUC7He:.CLIENT;qqf0n:.CLIENT;A8708b:.CLIENT;YcfJ:.CLIENT;VM8bg:.CLIENT;hWT9Jb:.CLIE
  NT;WCulWe:.CLIENT;szjOR:.CLIENT;JL9QDc:.CLIENT;kWlxhc:.CLIENT;qGMTIf:.CLIENT" data-new-gr-c-s-check-loaded="14.1067.0" data-gr-ext-installed>
   ▸<style data-iml="1657184331527">…</style>
   ▾<div jscontroller="HGv0mf" class="L3eUgb" data-sdd="200" data-sdh="150" data-sdssp="0" data-hveid="1"> flex
     ▸<div class="o3j99 n1xJcf Ne6nSd">…</div> flex
     ▸<div class="o3j99 LLD4me yr19Zb LS8OJ">…</div> flex
     ▾<div class="o3j99 ikrT4e om7nvf">
        <style data-iml="1657184331528">.om7nvf{padding:20px}</style>
       ▸<dialog class="spch-dlg" id="spch-dlg">…</dialog>
       ▾<form action="/search" method="GET" role="search">
         ▾<div jsmodel="sj77Re vWNDde P9Kqfe" jsdata="MuIEvd;_;CoXq+E">
           ▾<div jscontroller="cnjECf" jsmodel="QubRsd a4L2gc QwwFZb BFDhle gx0hCb TnHSdd icv1ie " class="A8SBwf" data-adhe="false" data-alt="false" jsdata="LVplcb;_;" jsaction="lX6RWd:w3Wsm
            c;DkpM0b:d3sQLd;IQOavd:dFyQEf;XzZZPe:jI3wzf;Aghsf:AVsnlb;iHd9U:Q7Cnrc;f5hEHe:G0jgYd;vmxUb:j3bJnb;R2c5O:LuRugf;qiCkJd:ANdidc;htNNz:SNIJTd;NOg9L:HLgh3;uGoIkd:epUokb;zLdLw:eaGBS;rcuQ
            6b:npT2md">
             ▸<style data-iml="1657184331528">…</style>
              <style data-iml="1657184331528">.CKb9sd{background:none;display:flex;flex:0 0 auto}</style>
             ▸<div class="RNNXgb" jsname="RNNXgb">…</div> flex
             ▸<div jscontroller="Dvn7fe" class="UUbT9" style="display:none" jsname="UUbT9" jsaction="mouseout:ItzDCd;mouseleave:MWfikb;hBEIVb:nUZ9le;YMFC3:VKssTb;vklu5c:k02QY;mMf61e:Mb6Xlc">…
              </div>
             ▾<div class="FPdoLc lJ9FBc">
               ▾<center>
...               <input class="gNO89b" value="Google Search" aria-label="Google Search" name="btnK" role="button" tabindex="0" type="submit" data-ved="0ahUKEwigufzatOb4AhWmSEEAHaHZBXsQ4dUDCA
                  0"> == $0
                  <input class="RNmpXc" value="I'm Feeling Lucky" aria-label="I'm Feeling Lucky" name="btnI" type="submit" jsaction="trigger.kWlxhc" data-ved="0ahUKEwigufzatOb4AhWmSEEAHaHZBXsQ
                  19QECA4">
                </center>
              </div>
            </div>
            <div style="background:url(/images/searchbox/desktop_searchbox_sprites318_hr.webp)"> </div>
          </div>
         ▸<div id="tophf">…</div>
        </form>
      </div>
     ▸<div class="o3j99 qarstb">…</div>
     ▸<div class="o3j99 c93Gbe">…</div>
```

Issues

The HTML DOM is always tree-structured - which is allowed by the structure of the HTML document.

This is cool because we can traverse trees fairly easily. Unfortunately, just because it is easy does not necessarily mean it will be quick.

Nowadays, DOM trees are huge.
Since we are being pushed towards dynamic web apps **(Single Page Applications - SPAs)** more and more, we need to modify the DOM tree instantly and often.

This presents a real performance and development pain.
Consider a DOM made of thousands of divs.

Remember, we are modern web developers, our app is very SPA! We have lots of methods that handle events - clicks submits, type-ins…

This presents **two problems**:

- It’s hard to manage. Imagine that you have to tweak an event handler.
  If you lost the context, you have to dive deep into the code to even know what’s going on, which is both time-consuming and puts you at a high risk of bugs.

- It is inefficient. Do we need to do all these findings manually?
  Maybe we can be smarter and figure out in advance which nodes will need to be updated?

React comes with a helping hand.

The solution to problem 1 is declarations.
Instead of low-level techniques like traversing the DOM tree manually, simply declare what a component should look like.

React does the low-level job for you - the HTML DOM API methods are called under the hood.

React doesn't want you to worry about it - eventually, the component will look like it should.

But this doesn't solve the performance issue.
And this is exactly where the Virtual DOM comes into action.

Virtual DOM

First of all - the Virtual DOM was not invented by React, but React uses it and provides it for free.
The Virtual DOM is an abstraction of the HTML DOM.
It is lightweight and is detached from the browser-specific implementation details.

Since the DOM itself was already an abstraction, the virtual DOM is, in fact, an abstraction of an abstraction.

Perhaps it's better to think of the virtual DOM as React's local and simplified copy of the HTML DOM.

It allows React to do its computations within this abstract world and to skip the "real" DOM operations, which can often be slow and browser-specific.

There's no big difference between the "regular" DOM and the virtual DOM. This is why the JSX parts of the React code can look almost like pure HTML:

```
const CommentBox = () => {
    return (
      <div className="commentBox">
        Hello, world! I am a CommentBox.
      </div>
    )
};
```

In most cases, when you have an HTML code and you want to make it a static React component, you must only do the following:

- Return the HTML code.
- Replace the class attribute name to className - because "class" is a reserved word in JavaScript.

There are additional differences as well, which we will learn about in future lessons.

You are now one step closer to start using REACT!

Good luck!