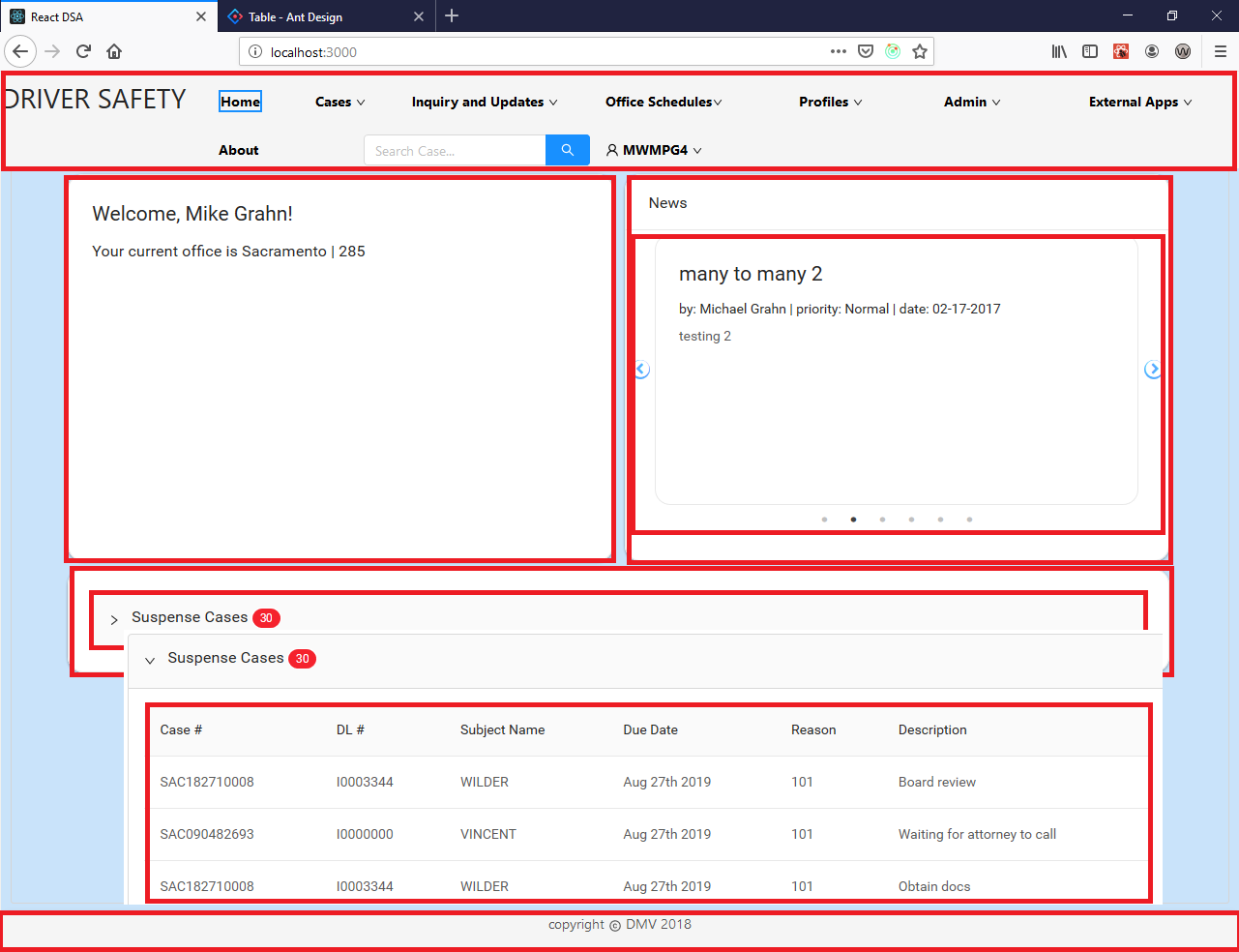
REACT DRIVER SAFETY APPLICATION USEFUL INFORMATION



React application is basically an arrangement of individual components. All the highlighted parts in the above picture are each a component, that are arranged in one high level “App” component placed in a “Root”.

Important Terms to understand:

**DOM:** DOM stands for ‘Document Object Model’. In simple terms, it is a structured representation of the HTML elements that are present in a webpage or web-app. It contains a node for each HTML element present in the web document. It is very useful as it allows web developers to modify content through javascript, also it being in structured format helps a lot as we can choose specific targets and all the code becomes much easier to work with.

**Updating DOM:** If you know a little about javascript then you might have seen people making use of ‘getElementById()’ or ‘getElementByClass()’ method to modify the content of DOM. Though doing thinks like this is not a problem and it works fine, but consider a case where we have a DOM which contains nodes in a large number and also all these web elements have different styling and attributes.

**Virtual DOM:** In React, Virtual DOM exists which is like a lightweight copy of the actual DOM. So for every object that exists in the original DOM there is an object for that in React Virtual DOM. It is exactly the same, but it does not have the power to directly change the layout of the document. Manipulating DOM is slow, but manipulating Virtual DOM is fast as nothing gets drawn on the screen.

**How Virtual DOM helps React?**  
Each time we change something in our JSX file, all the objects that are there in the virtual DOM gets updated. Though it may sound that it is ineffective but the cost is not much significant as updating the virtual DOM doesn’t take much time. React maintains two Virtual DOM at each time, one contains the updated Virtual DOM and one which is just the pre-update version of this updated Virtual DOM. Now it compares the pre-update version with the updated Virtual DOM and figures out what exactly has changed in the DOM. This process is known as ‘diffing’. Once React finds out what exactly has changed then it updated those objects only, on real DOM. This significantly improves the performance and is the main reason why Virtual DOM is much loved by developers all around.

**What are Props and State?**

Props and State are the two types of data that control a component.

**Props**

Props are short for *Properties*. The simple rule of thumb is props should not be changed. In the programming world we call it “*Immutable*” or in simple english “Unchangeable”.

Props are Unchangeable — Immutable

Components receive props from their parent. These props should not be modified inside the component. In React the data flows in one direction -> From the parent to the child.

**Props essentially help you write reusable code.**

# State

State works differently when compared to props. State is internal to a component, while props are passed to a component.

State can Change — Mutable

In english the ‘state of a being’ refers to the physical condition of a person, and it is a mere state, which changes over time. Similarly state in React is used within components to keep track of information.

**Anytime there is data that is going to change within a component, state can be used.**

**React state is stored locally within a component. When it needs to be shared with other components, it is passed down through props. In practice, this means that the top-most component in your app needing access to a mutable value will hold that value in its state. If it can be mutated by subcomponents, you must pass a callback to handle the change into subcomponents.**

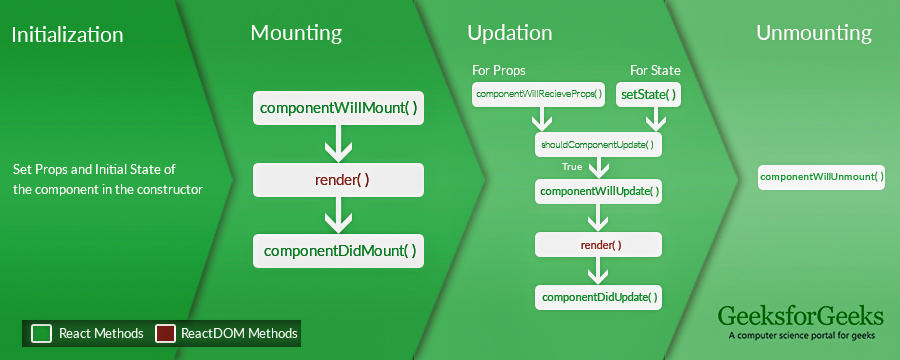
**When using Redux, state is stored globally in the Redux store. Any component that needs access to a value may subscribe to the store and gain access to that value. This centralizes all data but makes it very easy for a component to get the state it needs, without surrounding components knowing of its needs.**

**More on Props & State:**

<https://codeburst.io/props-and-state-in-react-native-explained-in-simple-english-8ea73b1d224e>

# <https://spin.atomicobject.com/2017/06/07/react-state-vs-redux-state/>

# COMPONENTS:



A React Component can go through four stages of its life as follows.

* **Initialization:** This is the stage where the component is constructed with the given Props and default state. This is done in the constructor of a Component Class.
* **Mounting:** Mounting is the stage of rendering the JSX returned by the render method itself.
* **Updating:** Updating is the stage when the state of a component is updated and the application is repainted.
* **Unmounting:** As the name suggests Unmounting is the final step of the component lifecycle where the component is removed from the page.

Useful Links:

<https://www.geeksforgeeks.org/reactjs-lifecycle-components/>

<https://www.freecodecamp.org/news/all-the-fundamental-react-js-concepts-jammed-into-this-single-medium-article-c83f9b53eac2/>

<https://github.com/uanders/react-redux-cheatsheet/blob/master/article/react-redux-concept-workflow.md>

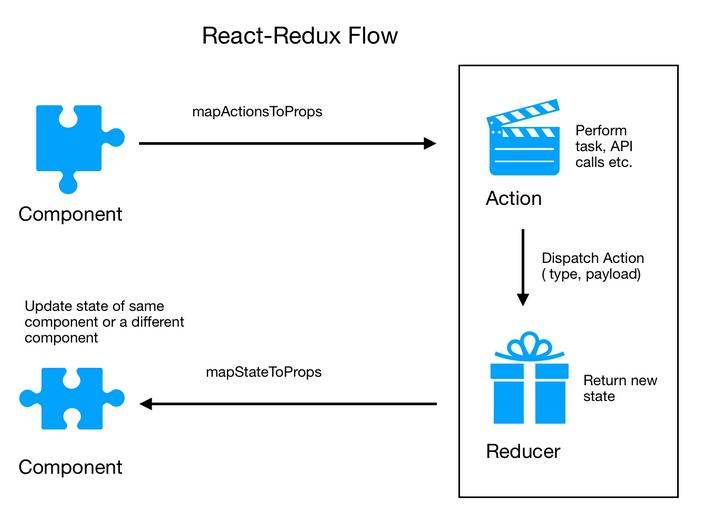
# A Note on Smart vs. Dumb Components

React/Redux applications are made up of ‘smart’ and ‘dumb’ components. Dumb components are intended to be passed exactly the data they need to render their piece of the puzzle, and no more. They should be ignorant (‘dumb’) to the rest of the world. In general, any callbacks they need should be passed down to them from another component. Static pages are obvious candidates for dumb components: They likely have no business knowing when state is updated in the rest of the application.

Smart components then, are sensitive to state changes in the rest of the application. They can initiate changes themselves, or can listen for changes produced by other smart components. What this means technically is that smart components have access to the single global state object (in our app, we use Redux to manage this single global state object).

What’s a Redux?

Redux is an architecture pattern and a library which lets us manage our application’s state in a single object.

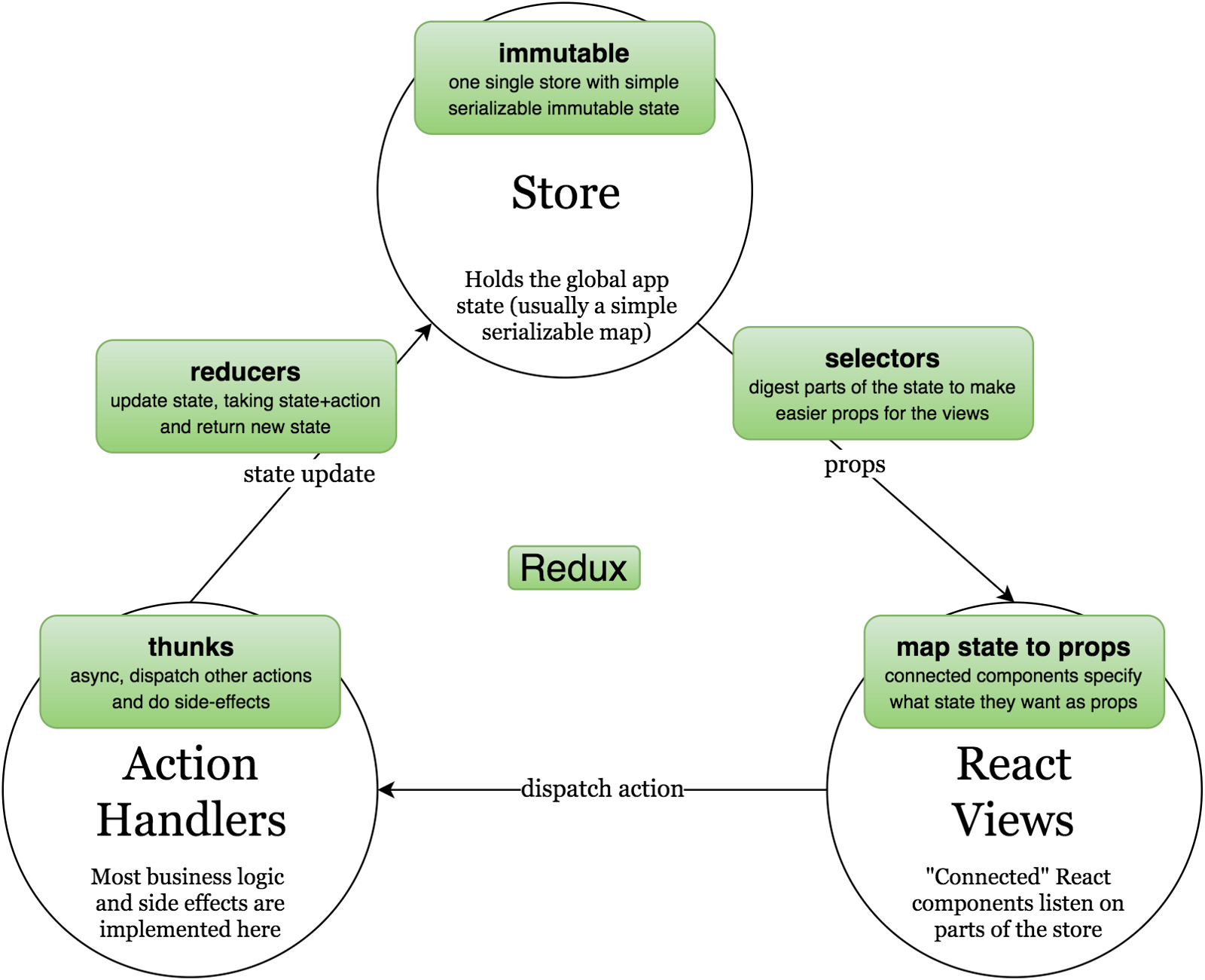


Useful Links:

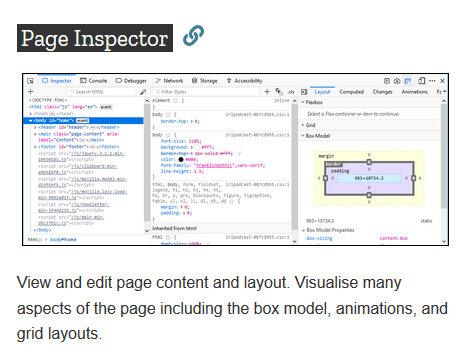
<https://quickleft.com/blog/redux-plain-english-workflow/>

<https://hackernoon.com/https-medium-com-heypb-react-redux-workflow-in-4-steps-beginner-friendly-guide-4aea9d56f5bd>

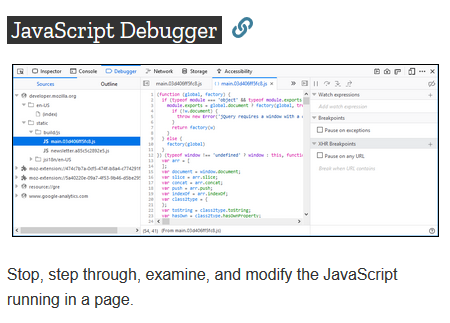
# React – Redux Illustration:

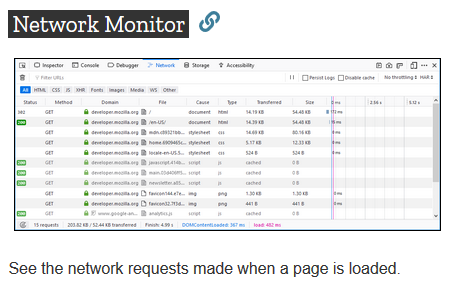


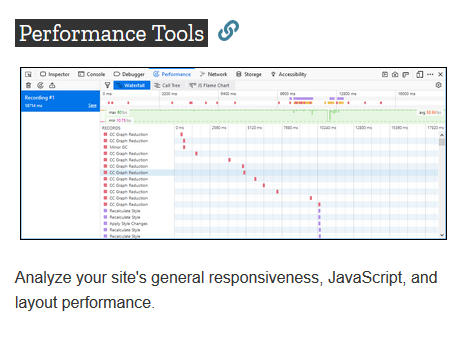
**Firefox dev tools:**

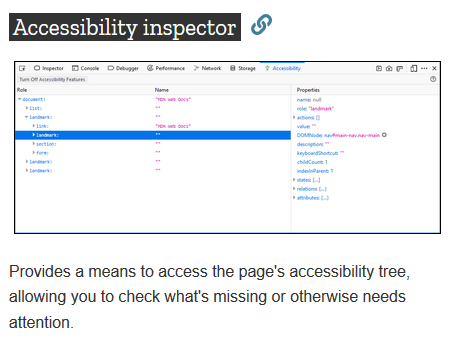








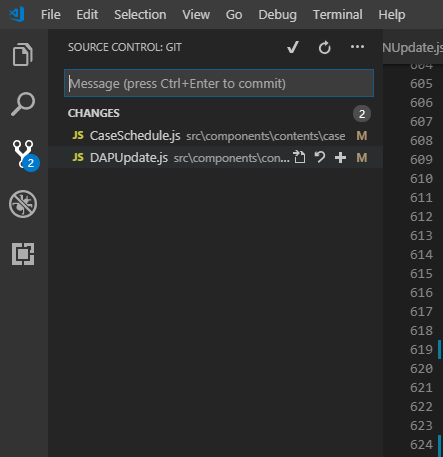




**Pushing DSA Web Client changes to TFS:**

To push changes to the TFS,

1. We need to “stage” the changes first.
2. We need to “sync” – calls “Pull” and “Push” from TFS to sync the code.



**Deploying Web Client to Server:**

To generate the deployment files, we need to run a script called build.js. To do this using NPM,

1. Run “npm run-script build” in the Terminal. Build.js file has build settings configured in it.
2. After the build is completed, the files will be generated in the “build” folder in the DSA Web Client repo.
3. Copy ALL the files from the “build” folder and paste them in the “DSA WebClient” folder on the server and refresh the server.

**Third party, Open sourced react component libraries used in DSA Web Client:**

[**Axios**](https://github.com/axios/axios)**,**

[**Ant Design**](https://ant.design/)**,**

[**PrimeReact**](https://www.primefaces.org/primereact/#/)**,**

[**fullCalendar**](https://fullcalendar.io/)**,**

[**React Datepicker**](https://reactdatepicker.com/)**,**

[**moment**](https://momentjs.com/)**,**

[**cloneDeep**](https://www.npmjs.com/package/clone-deep)