**React Notes – Wasif Hossain**

**What is React?**

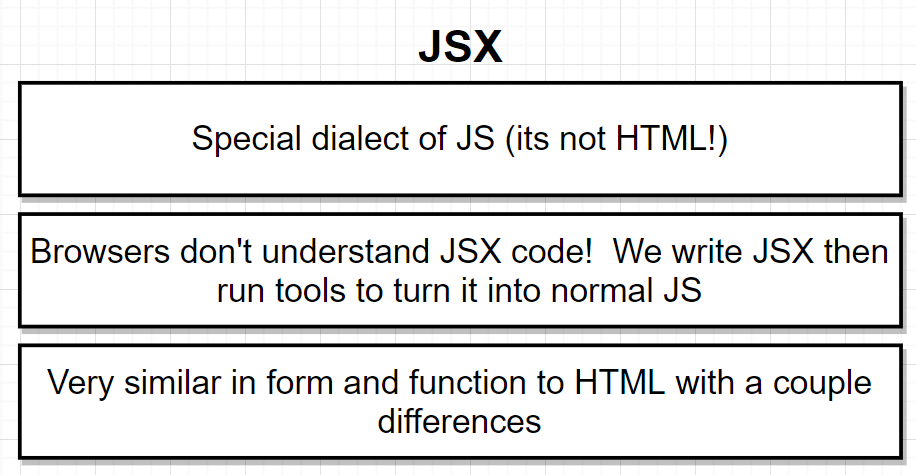
React is a javascript library/ framework that is responsible to show content (HTML) to users and respond using event handlers for user interaction and manipulation.

**What is class?**

React components are made using either javascript functions or classes and present some content to the screen.

**What is JSX?**

Javascript Expression which looks like HTML but can be placed as Javascript code. It determines the content of our react app just like normal HTML.



**Syntax**

* Use double quotations “ “ to indicate a string such as className = “label”
* Any non-JSX property, use single quotes such as backgroundColor: ‘blue’
* Not allowed to take a javascript object and reference it inside JSX especially where text is shown.

**HTML vs JSX**

HTML: <div style = “background-color: red;”> </div>

JSX: <div style={{ backgroundColor: ‘red’}}> </div>

* Outer bracket {} of the JSX div tag refers to a javascript variable
* Inner bracket {} of the JSX div tag refers to the javascript object
* Where the keys of the object reference a property we would want to style and values indicate the value for that particular styling

**An event handler?**

Event handlers are used to detect user interaction and respond to it, update our application or component when a change occurs that is triggered by a user.

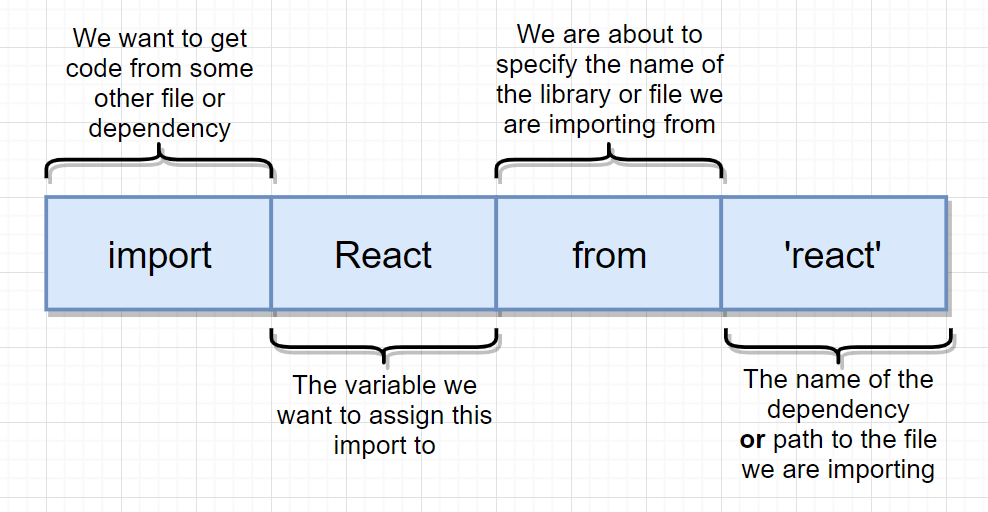
**React & ReactDOM**

* React knows what a component is and how to make components work together
* ReactDOM knows how to take a component and make it show up in the DOM

**What is babel**

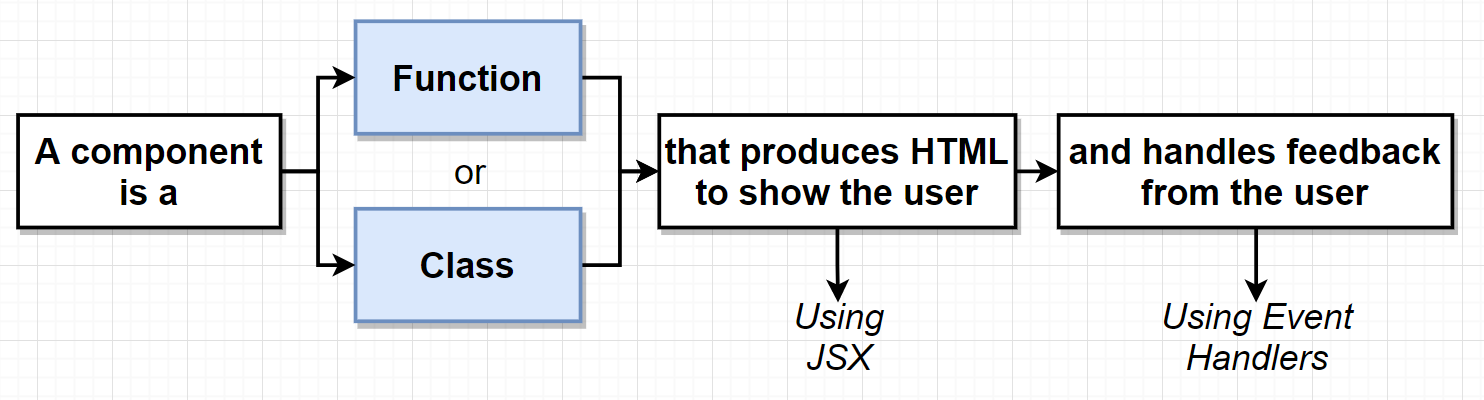
* Babel is a command line tool which gets any version of JS and creates a newer version that is compatible with browsers

**Import statements?**

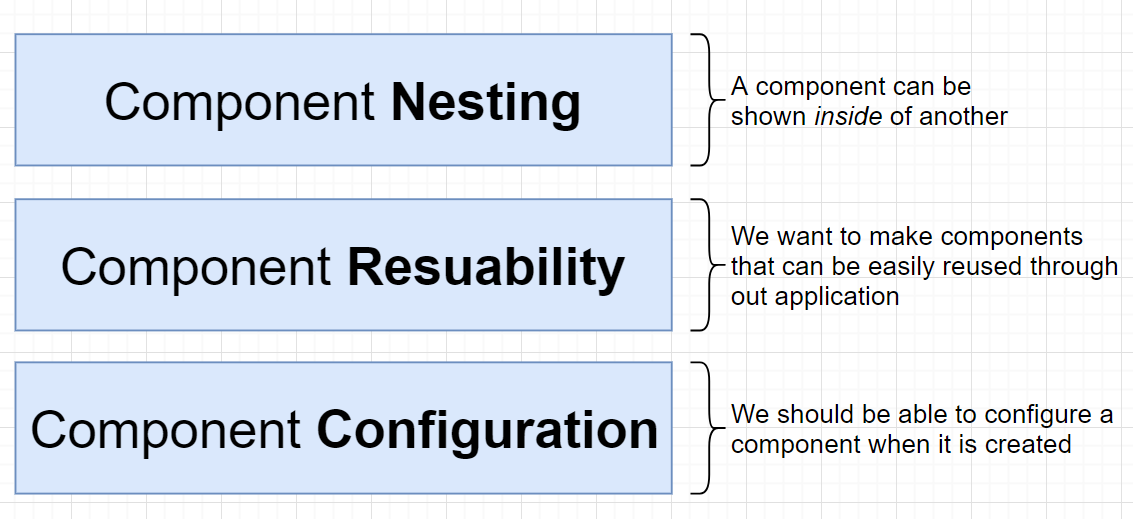


**Components**

Components are building blocks of React applications and returns an element that describes the user interface.

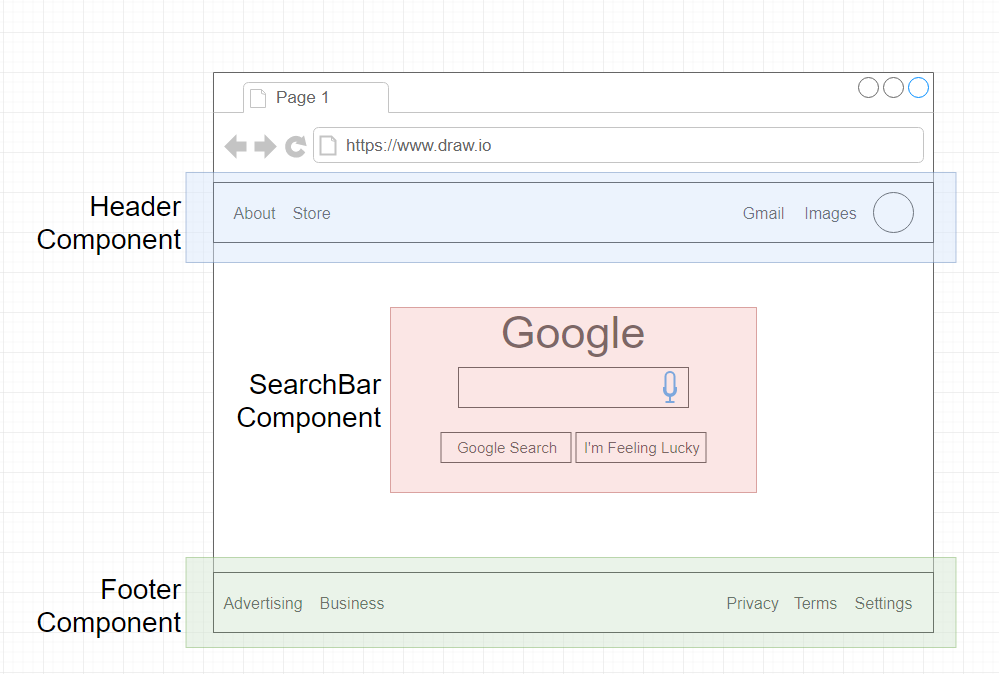
****

**Three tenets of components**

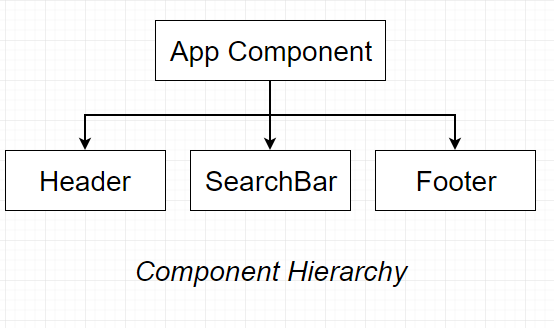
****

**Components**

* Components are suppose to show content to the user
* React to user interaction



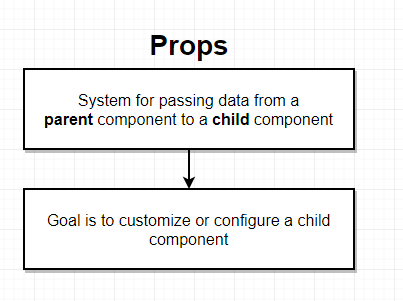
**Component Hierarchy**

****

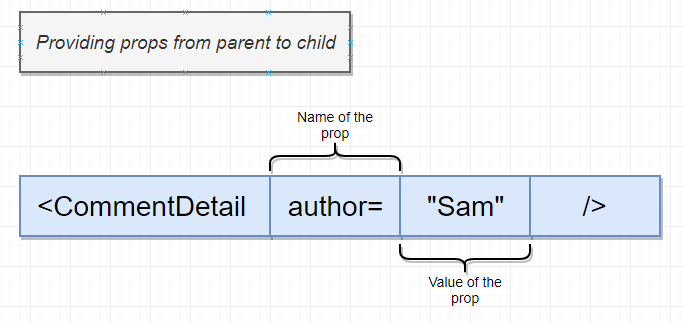
* App component is the parent component
* Header, search bar and footer are child components

**Props**

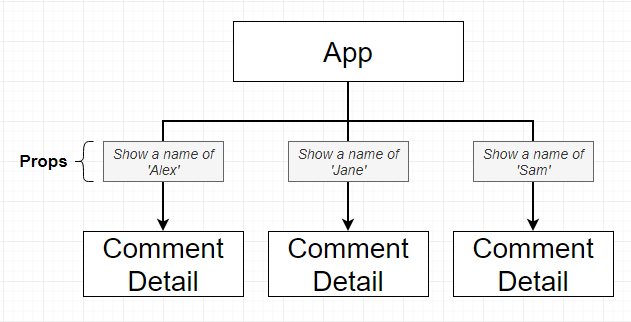
* Props means properties
* Communicate data from a parent component to a child and customising a child.
* However, a child cannot pass data to the parent component with props system directly



* App component will show instances of comment detail and it will pass a bit of configuration to each of those components

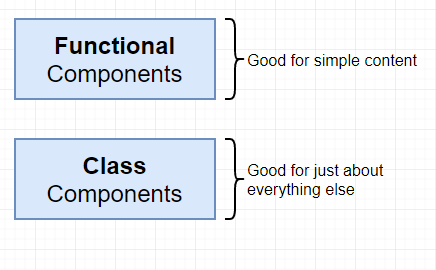


Passing props from parent component to children



**When to use functional or class based components?**

* Functional components are good for displaying simple content without a significant amount of logic (create some JSX and returns it)
  + Functional components cannot handle async code.
* Class based components are for more complex content that involves event handlers or manipulation of data and logic

****

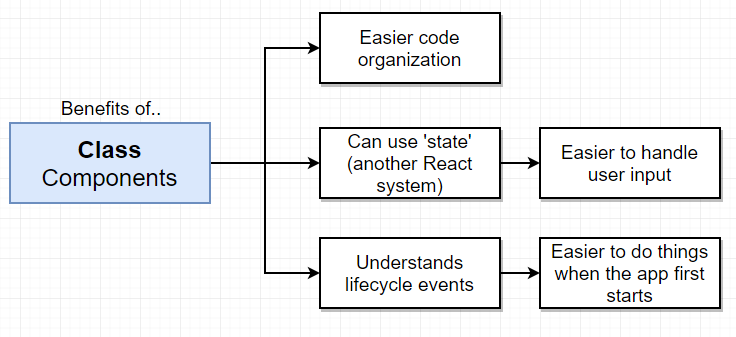
**Class based components**

* Easier code organisation
* Can use state → easier to handle user input
* Understands lifecycle events → easier to do things when the app first starts

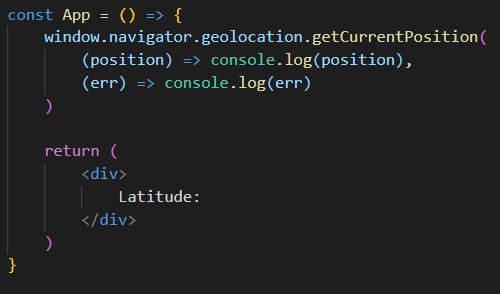
**Rules of class components**

* Must be javascript class
* Must extend subclass React.component (prototypal inheritance)
* Must define a render method that returns some amount of JSX

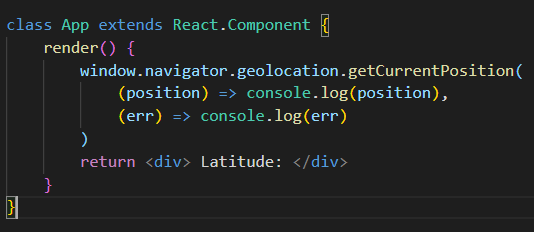
**Benefits of class components**



**React Function Code**



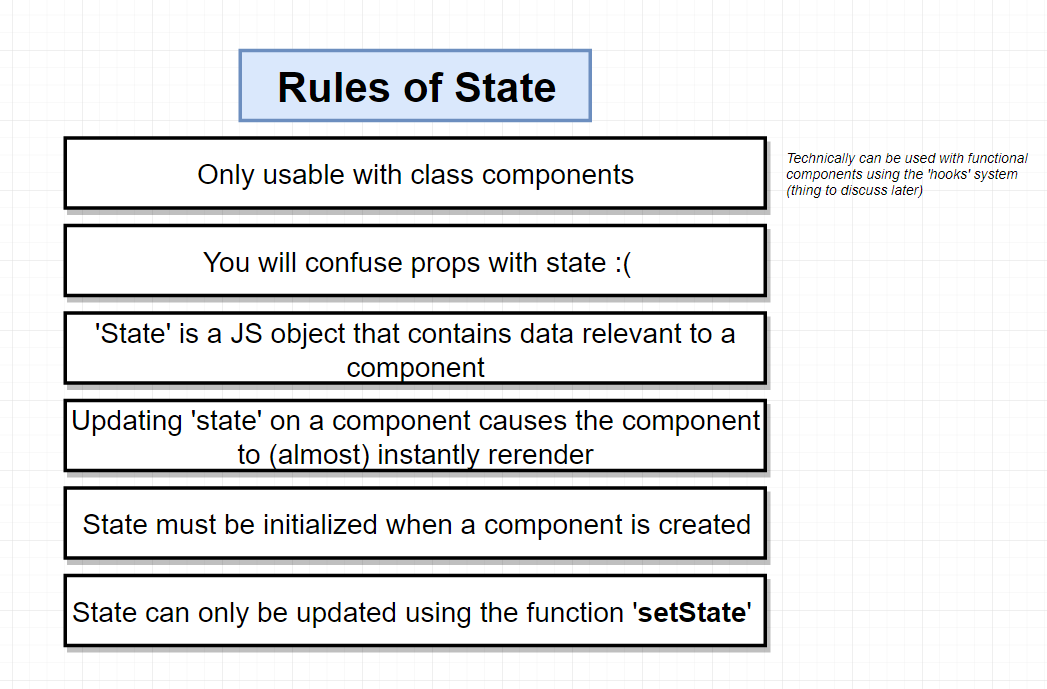
**React Component Code**



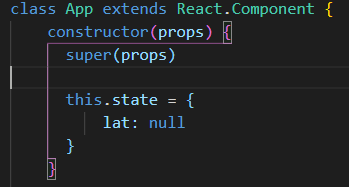
* Has a method assigned to it such as the render
* React expects that our class based component has many other methods attached to it and we borrow all these other methods from this other class called React.Component

**State**

* Use state if we expect data to change/update overtime

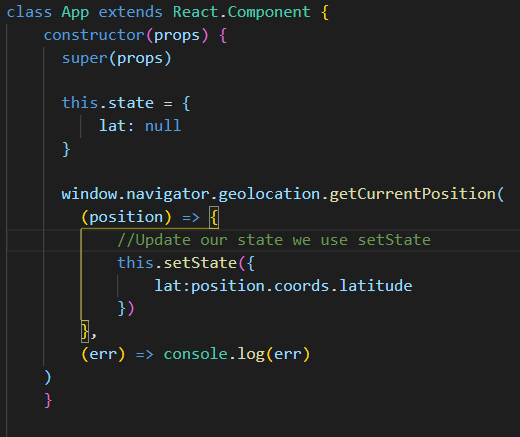
****

* Have to initialise the state object within the constructor function



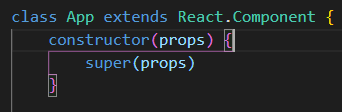
* Initialises the state object with a key value ‘lat’ and value of ‘null’ as the latitude value is unknown

**Updating State**



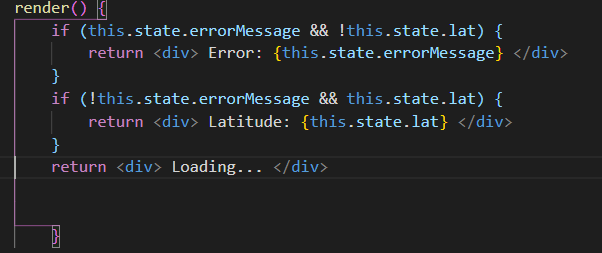
* Updated function using setState which passes an object that would update the property you would want to T

**Constructor**

****

* Constructor function is called first and creates an instance of the component
* App is borrowing functionality from react component and the base class has a constructor function of its own that has some setup for a react component.
* Therefore when a constructor function is defined in the app class we are overriding the constructor function that is defined in react.component
* Super(props) allows us to call setup functions in react.component and therefore uses ‘super’ to reference the parent’s constructor function

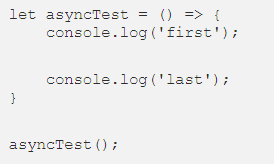
**Conditional Rendering**

****

* Depending on the props and state of our component we return different jsx

**Synchronous Code**

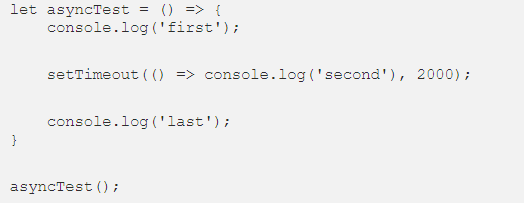
* Synchronous code waits for previous event or an event to be called then continues to execute



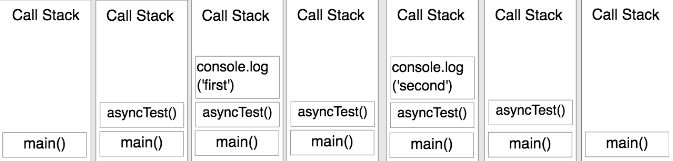
* In this example the output is first, last

**Asynchronous Code**

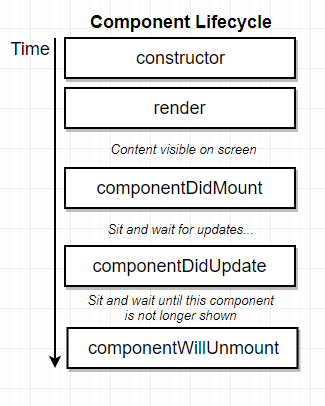
* Asynchronous code involves call back functions



* In this example we get the output first, last, second
  + This is because the call back function works like a stack call (LIFO - Last in First Out)
  + Where the function that is added to the top of the stack is ran first and if the function calls another function then it is added to the top of the stack then runs it. Only the function at the top of the stack is the one that can be run

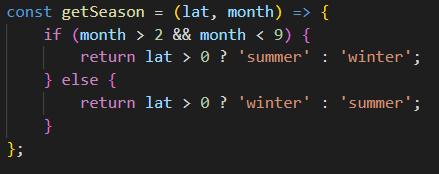


**Component Lifecycle**

* Constructor - good place to do one-time setup
  + According to the community of React, it is best not to do data loading in the constructor function but within componentDidMount
* Render - avoid doing anything besides returning jsx
* componentDidMount - Good place to do data loading
* componentDidUpdate - Good place to do more data-loading when state/ props change
* componentWillUnmount - Good place to do cleanup especially for non-React stuff

**Ternary operators**

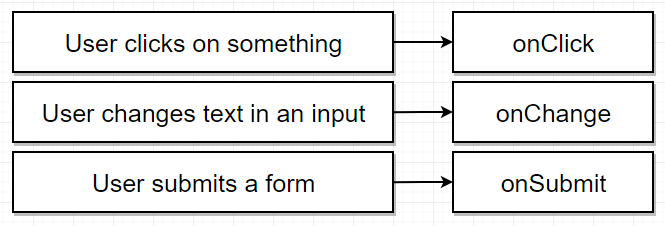
* This is an expression which allows you to have a value that can be assigned to a variable based on a condition
  + Condition ? true : false



* In this it is stating that if the latitude value is greater than 0 then return summer else winter

**Event Handlers**

Find an element to watch for some event and add a prop of whichever event we want to watch for (click, change or submit) and we pass that prop a callback and that callback will be invoked whenever the event occurs

****

* These are important property names which execute a callback function
* Convention name for these callback functions are
  + onClick : onInputClick
  + onChange: onInputChange
  + onSubmit: onInputSubmit
* Syntax



* Alternate syntax (preferable for one line code and does not require to create a separate callback function)



**Controlled vs Uncontrolled Components**

* Uncontrolled
  + Stores information on the DOM (Document Object Model) or HTML
  + Uses references to access values from the DOM
* Controlled
  + Storing information inside components on a state property
  + Uses state and is easily accessible from the React Component

**‘This’ in javascript**

* ‘This’ is a reference back to the class itself

**Bind in javascript**

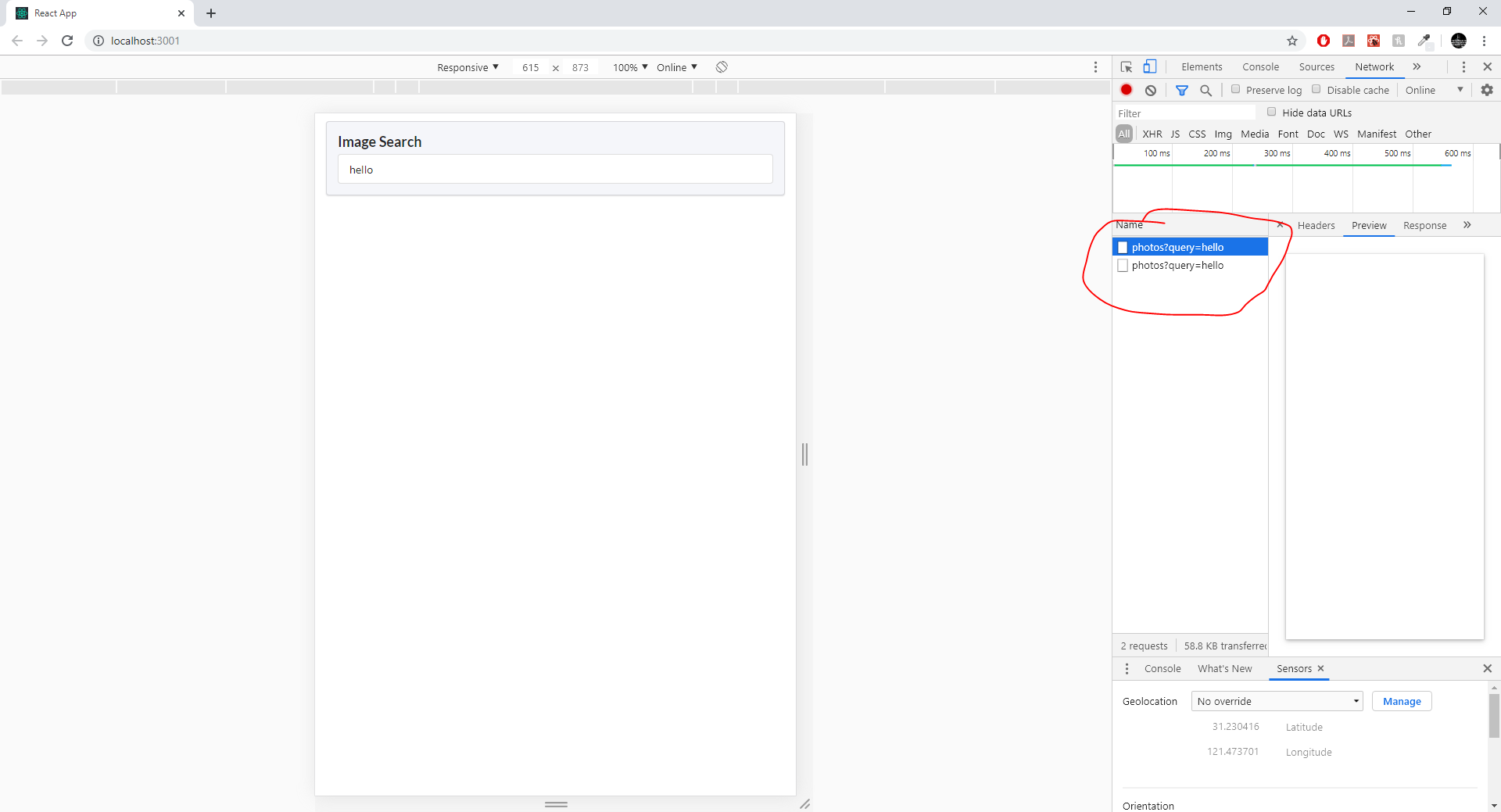
* Produces a new function which is fixed with the correct value with “this” or context (instance of the class)
* Arrow functions automatically bind the value of ‘this’

**Fetching Data**

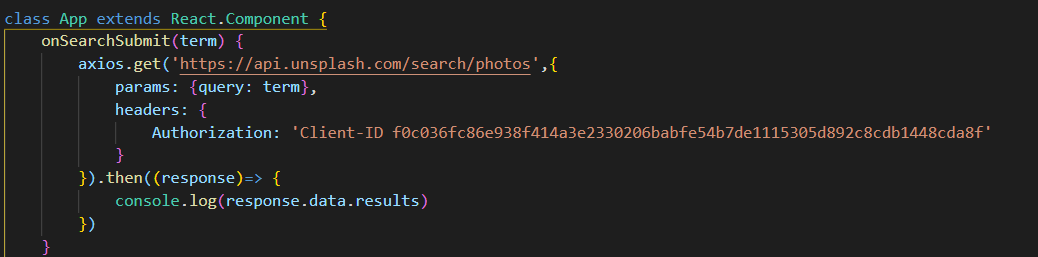
* Create an Ajax request through an app component to the API
* The API will send a list of json records (contains information)
* React is responsible for showing content/ data to users and handling user interaction
* Handling http request is done through separate piece of code or Ajax

**Axios**

* Axios is a 3rd party package for managing https requests
  + Has already a lot of code that is reusable so makes you write less code



* When making a request there will be two requests due to cross origin resource sharing



* Whenever a request is made with axios, it returns an object called a promise which provides a notification when a network request is completed
* .then() function returns something anytime we get a response from an asynchronous request (used when working with a promise)

**Alternative way to handle async requests**

****

**Fetch**

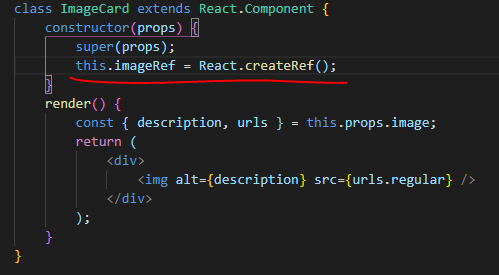
* Fetch function is built into modern browsers
* Basic and lower level function to use to fetch data
  + Write more code

**Javascript functions**

* Map function - Iterate over an array and returns a brand new array and it will take each of the original values and modify them

**React Refs**

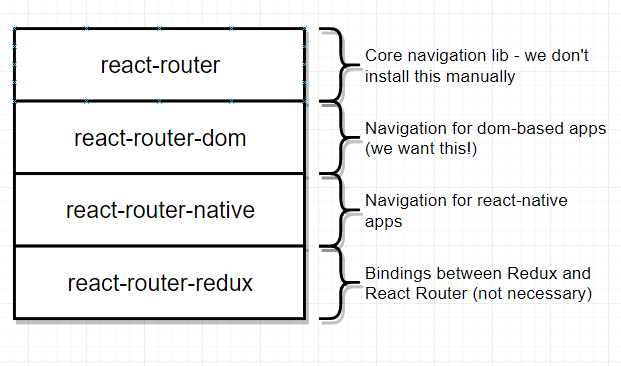
* Gives access to a single DOM element
* We create refs in the constructor, assign them to instance variables then pass to a particular JSX element as props



* this.imageRef = React.createRef() → creates a reference which is assigned to an instance variable

**React Router**

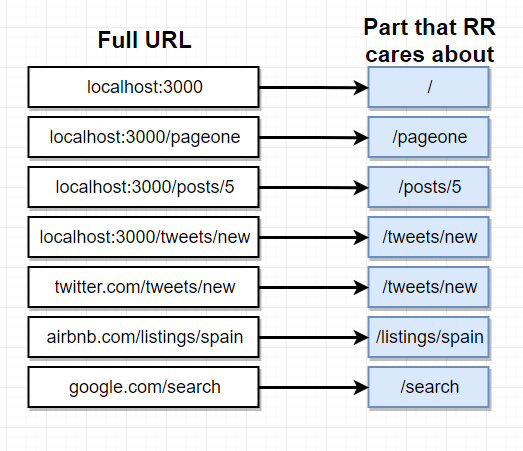
* Npm install --save react-router-dom
* React-router is a poor library



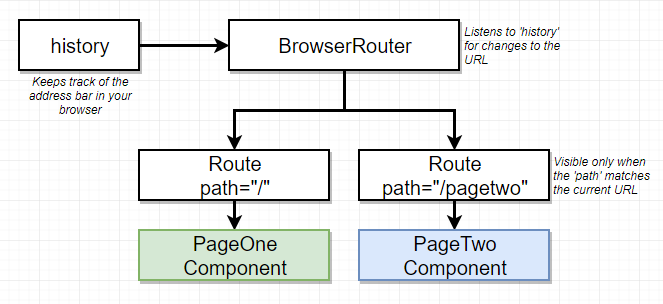
* Based on the URL, seeing different content on the screen

**How React Router works?**

* React router does not care about the entire url (localhost:3000)
  + Only cares about the url after the domain definition



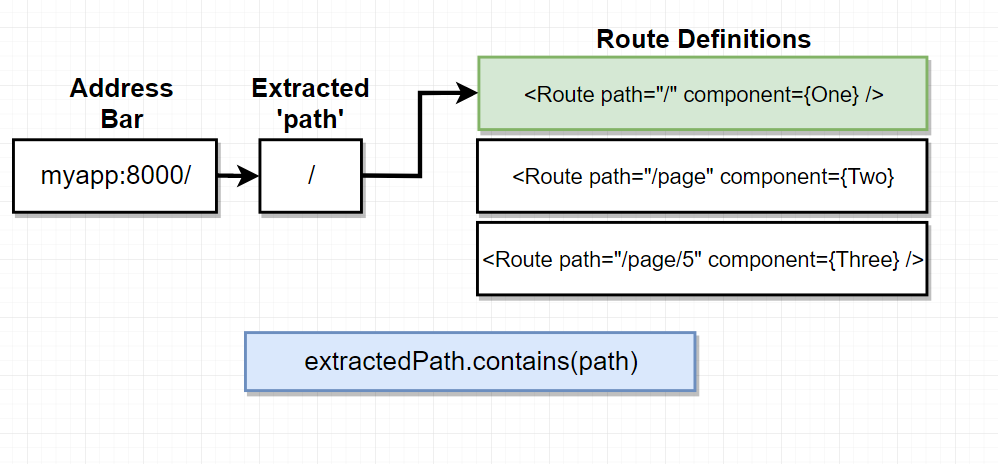
**How React Router uses it ?**



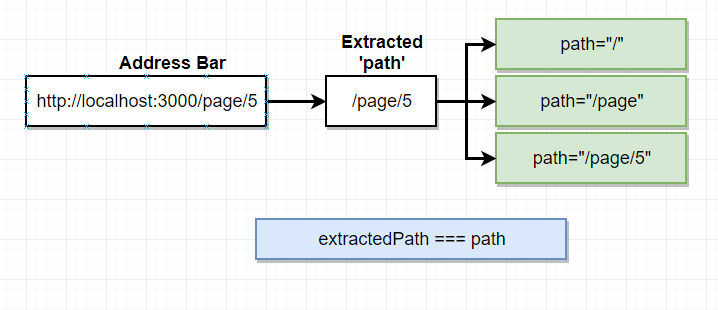
* When application is created, it creates an instance of the browser router component
* Browser router component internally creates an object called history which looks at the URL in the address bar and extracts the section after the domain name
* History object communicates that path down to browserrouter and browserrouter communicates that path to both the route components.
* The route components would then decide to show themselves depending on the path and site of URL and path property it was passed when created
* Inside react router application, can have multiple router components which match url and show to the user
  + Deeply nesting routes, can customise some parts of how the app looks and don’t have to pass down some deep configuration through redux or props to configure how a deeply nested child component renders itself

**How the path extraction works**

* Takes the string of the extracted path URL
* Runs a contains method to the path of each the routed components for comparison

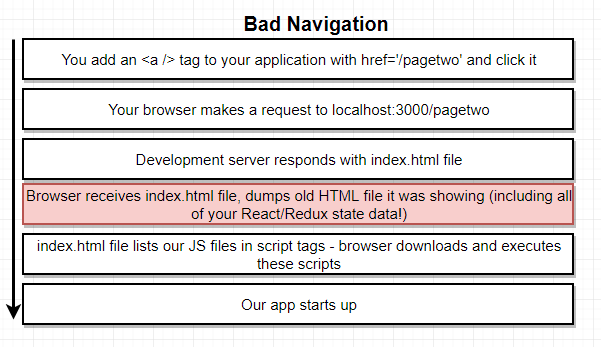


* For example in this scenario, it checks for the extracted path “/page/5” and compares against the paths and renders those pages on the screen and therefore contains the strings of the extracted path
* Exact is equivalent to saying “true”
  + Causes React Router to change the rule slightly to extractedPath === path where
    - “/” !== “/pagetwo”



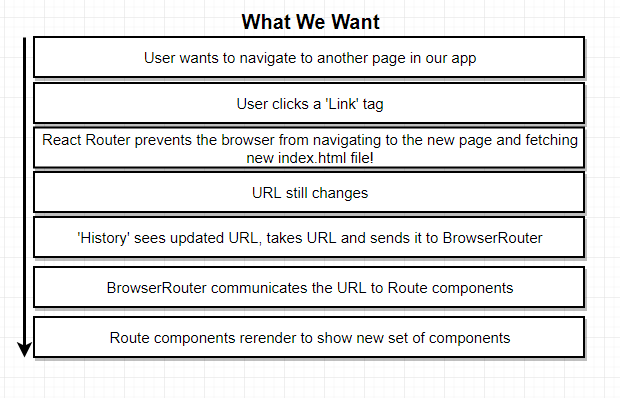
**Navigation using React Router**

* Users can navigate through the two routes by changing the address to the address bar (**not best to use anchor tags!** )
  + This is because it dumps all your data and variables (wiped)
* Use Link Tag

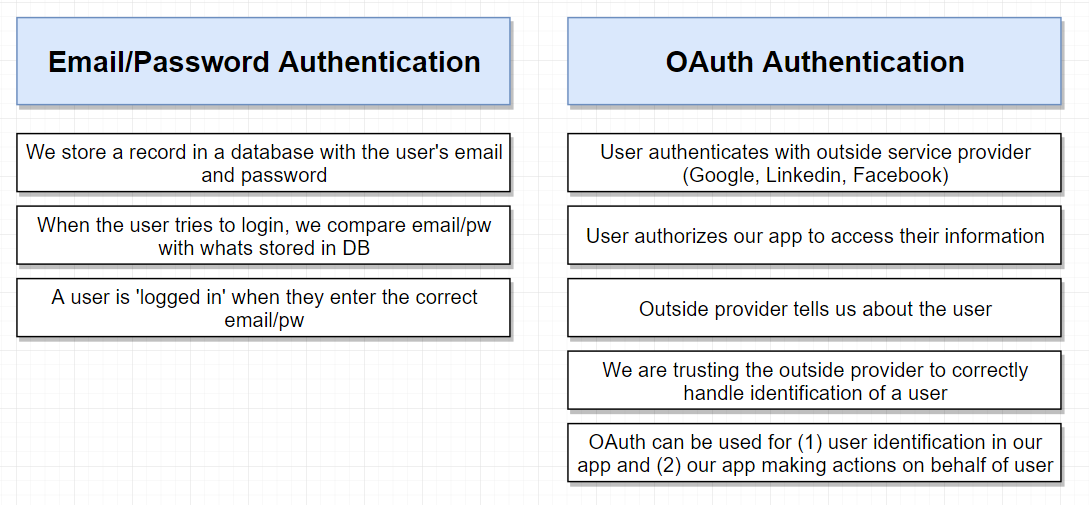


**Link Tag**

* Not dumping react data but just showing different set of components on the screen (not requiring an additional request is required) and this is known as SPA (Single Page App), so only loading up a single HTML document and are making use of the same HTML document and just showing and hiding different sets of components based upon the URL
  + We are tricking the user thinking that they are going different pages but we are in fact showing and hiding different components

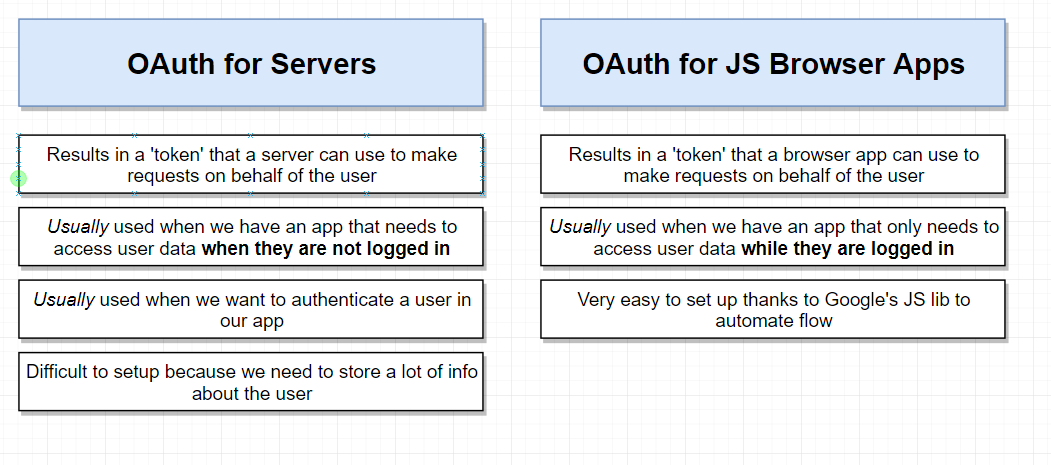


**O-Auth Authentication**

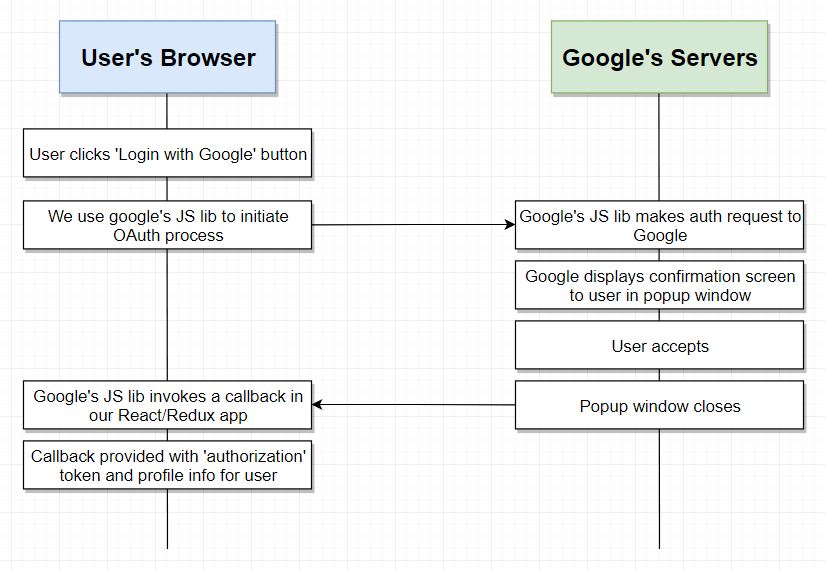


**O-Auth for JS Browser Apps**

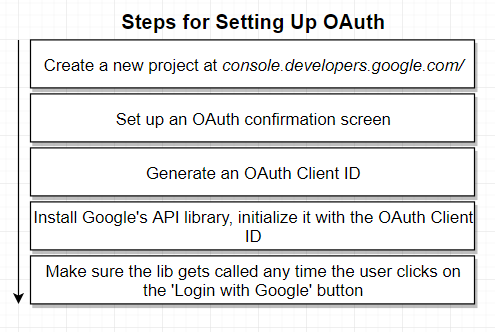
* Provide access to their information
* Receive a token and allow us to take action to the user without them being involved

****

**Flow**

****

**Steps for O-Auth**

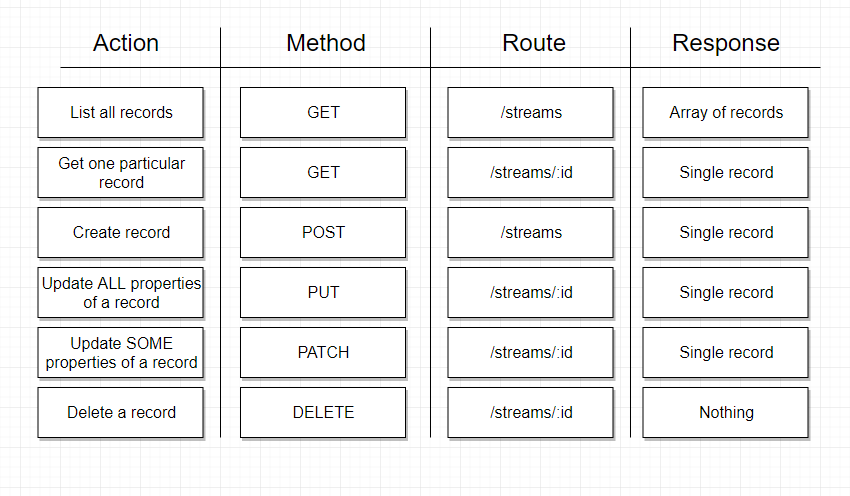
****

**What is proto?**

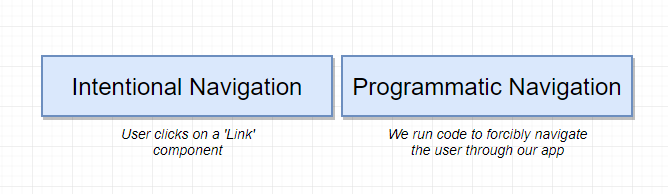
* Proto type property is how a javascript does inheritance between different classes

**REST Conventions**

* Predefined system for defining different routes on an API that work with a given type of records
* Standardised system of routes and request methods used to commit or operate different actions

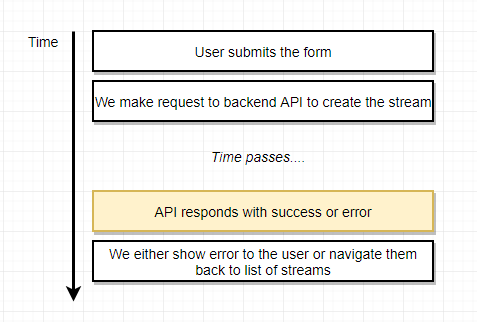


**Navigation**

****

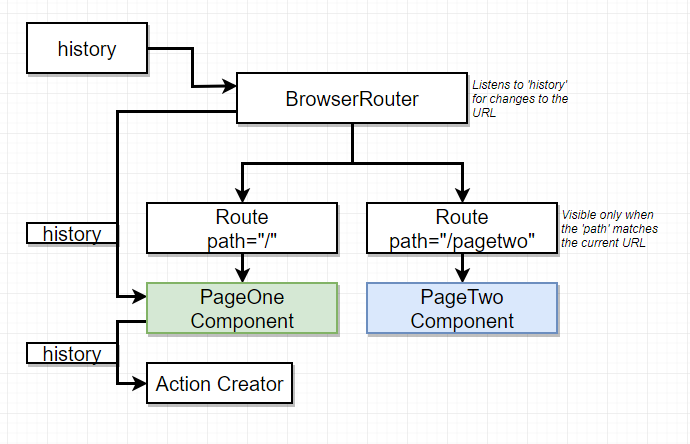
* Intentional Navigation is when a user clicks on a link component
* Run code in response to some event and will change the page that the user is looking at

**Goal of Programmatic Navigation**

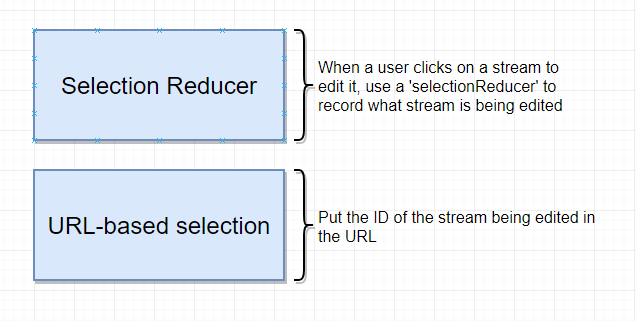
****

**Using non-react components to deal with history**

* Each time the component calls the action creator, it can pass along the history object inside the action creator

****

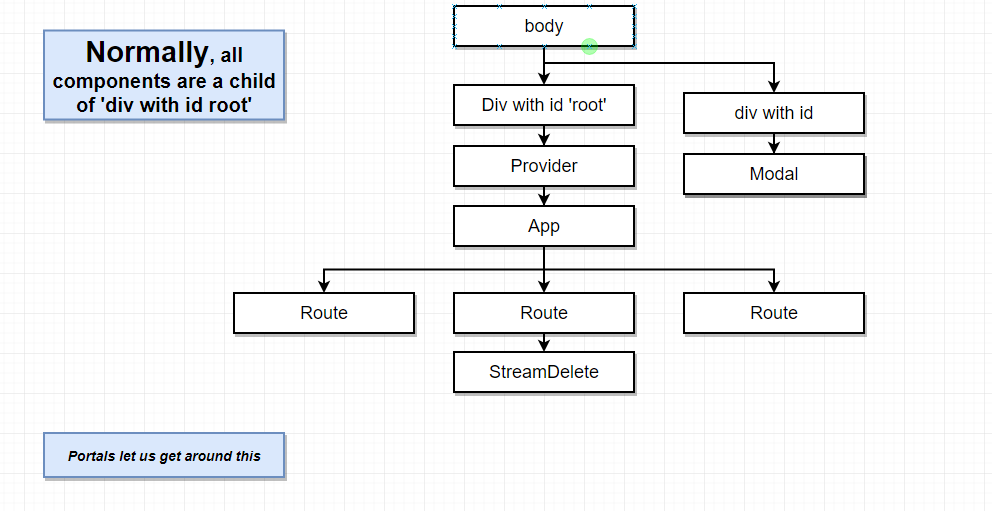
**Selection Reducer**

****

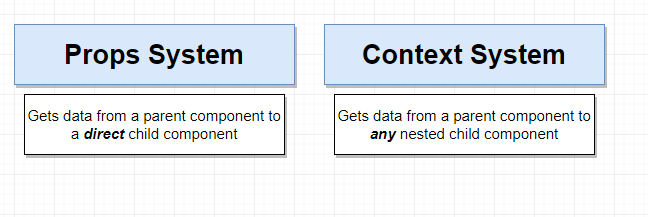
**Put vs Patch Request (Back-end)**

* Put request - whatever properties are contained within that request will replace all the properties inside the record you try to update, potentially deleting records (only id is immune to this)
* Patch request - Pass some properties in the body of the request that are suppose to be updated (not overriding) and therefore won’t drop off some properties

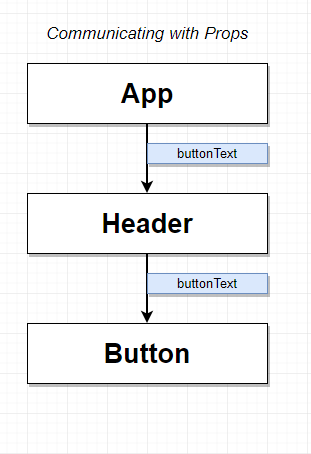
**React Portals**



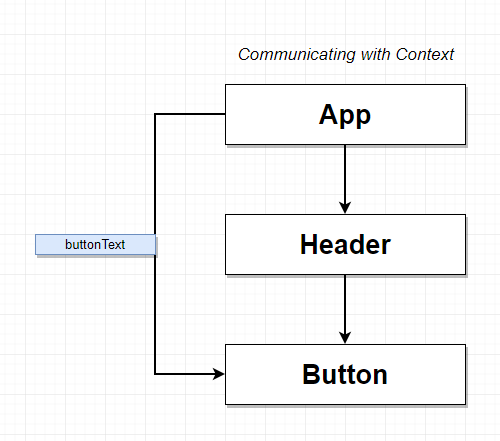
**Context System**



**Props System**

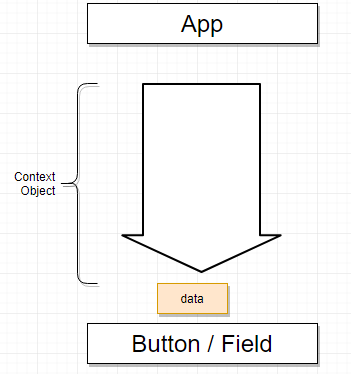
****

**Context System**

****

**Context System**

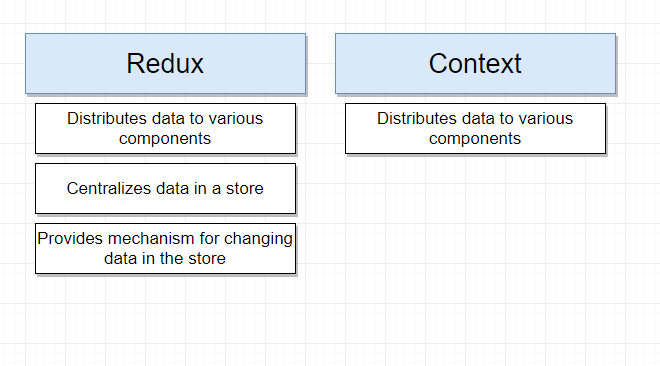
* There are two ways that you can get data flowing in/out of the pipe (context object)
* which is known as the context object (such as setting up a default value or create a provider object within the parent component)
* Then the data can be extracted out of the context object by referencing the this.context property or create a component called consumer from the nested child
* Create context object, define context type and get this.context

****

**Provider**

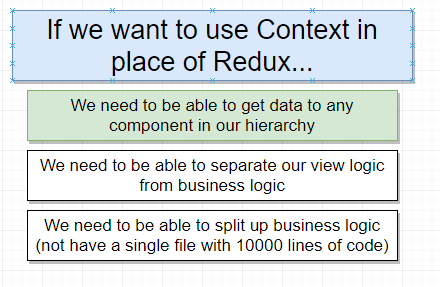
* Provider creates a separate type of information every single time you use it. There is a separate channel of information flowing down to a separate set of components.

**Redux Vs Context**

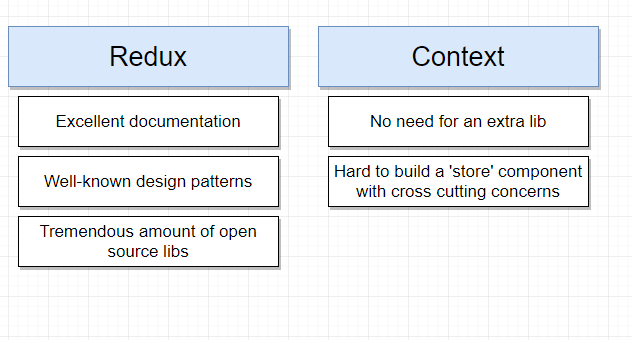


* Context by itself lacks a lot of other features with redux

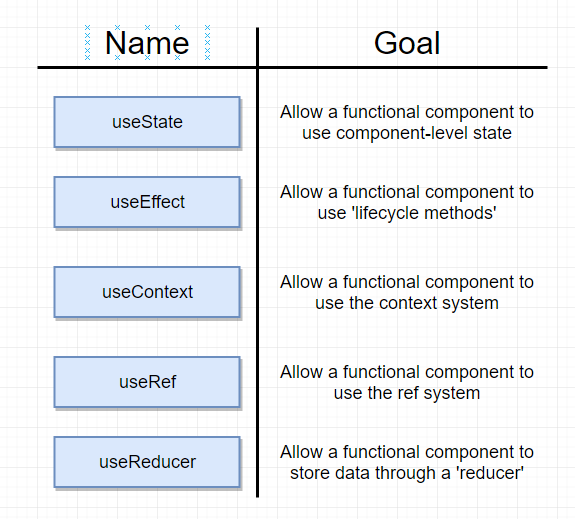
**Using Context To Replace Redux**

****

**Redux over Context**

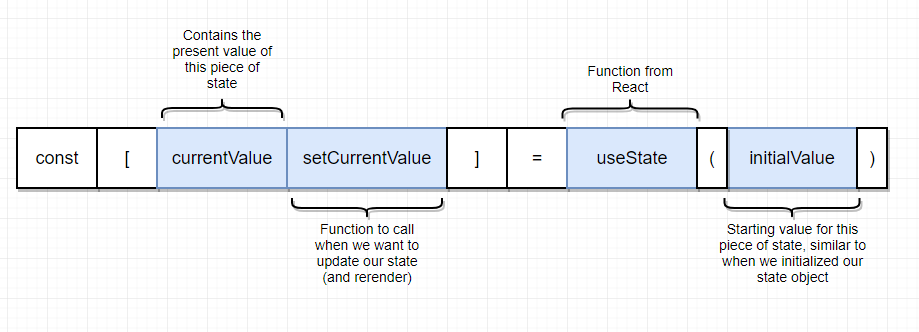
****

**React Functional Components**

****

**Functional Components Dissected**

* [ ] → Square brackets is destructuring the array

****