React

- Cheatsheet

Cheatsheet

Intermediate React

- **useState** container for state within component (instance props). Use a store for data.
- **useEffect** Update render callback; update when something happens - async events. It's scheduled and you don't know when its going to run
- **useContext** globall application state (globals) and make it available specifically
- **useRef** container for state reference outside of the application scope (modals, subscriptions) aka static class properties exists across class instances. Referring to dom elements within react application
- **useReducer** similar to filter, pass in state run it through reducer and return state. Good for testing. Decoupling state from display layer
- **Redux** good for managing global state. Use `useState` for inner-component state.
- **Object.assign** differential updating
- **useMemo** caching expensive computation preventing re-render only on change. For specific use cases. Have performance problems before using this
- **useCallback** implemented with useMemo. Takes in a function and a dependency, returns the same function so does not cause rerender.
- **memo** continue only when props change, for functional components
- **useLayoutEffect** similar to class component componentDidUpdate: happens immediately after your render finishes. Suitable for animation or measuring the dom
- **useImperativeHandle** very rare, useful for library authors. Exposes a function to the parent component.
- **useDebugValue** Add information readable by React Dev Tools in the DebugValueComponent. Console log for custom hook
- tailwind skipped
- **redux** :
  - state tree. React sends actions and reads state from redux.
  - useDispatch: query the storeh
- test:
  - test from the user perspective, button changed x to y
  - test features that are important to your app
  - test that fails on the bug, then make that test pass by fixing the bug.
  - No tests are better than bad tests (which take away confidence)
- Higher Order Component: reusable logic