REACT SO FAR

- React gives components for view
- Components:
    - are JSX transpiled to JS funcs to output HTML
    - are composable
    - can be class-based or function-based
        - info6250 only using function-based
    - Must be one containing element or fragment
        - children/descendants okay
        - not sibling elements

VIRTUAL DOM SO FAR

- Fast to change
- No visual changes
- Updates the real DOM if there ARE changes
- What React THINKS the actual DOM is
- Do not have React + non-React change DOM
    - Causes confusion between VDOM and DOM

CREATE REACT APP SO FAR

We are use Create-React-App (CRA) to make our apps

- Not required for React
- Handles webpack/babel
- Provides dev server (No normal server!)
    - You need to provide your own normal server
- Creates static files (only!) for eventual use
- Assumes SPA-only (no Progressive Enhancement)

MORE ABOUT COMPONENTS

- Class-based have state and "lifecycle methods"
    - But we aren't using class-based
- Function-based have **hooks**
    - Provide state (per component!)
    - Interact with rendering lifecycle
- Components are passed **props**
- Components props are often the state of ancestor (wrapping) components
- Props can only pass down (to descendants)
- Props can be **any** kind of JS value

MORE DEPTH

- Components can pass state to descendants as props
- Descendants can communicate "up" only via callbacks, which the ancestor must pass down
- Side note: YAGNI - don't add complexity initially, try to avoid it entirely

BASIC STATE

```
import React, { useState } from 'react';

const Counter = () => {
  const [count, setCount] = useState(0);
  return (
    <button onClick={ () => setCount(count+1) }>
      I am {count}
    </button>
  );
};
```

MORE STATE

```
const CompareTwo = () => {
  const [first, setFirst] = useState(0.1);
  const [second, setSecond] = useState(0);
  const winning = first > second ? "First" : "Second";
  return (
    <div> {winning} is ahead!
      <Thing
        value={first} onClick={() => setFirst(first+1) }
      />
      <Thing
        value={second} onClick={() => setSecond(second+1) }
      />
    </div>
  );
};
```

WHAT DID WE JUST SEE?

- You can call multiple `useState()`
- You can pass state in one component as prop to another
- Changing state causes a re-render

STATE EXERCISE

Rock - Paper - Scissors Game

- Rock beats Scissors
- Paper beats Rock
- Scissors beats Paper

Create a `<Game>` component that shows:

- A `<Choose>` until the user picks one
- After user picks, shows:
    - A `<Choice>` passed what they picked
    - A `<Choice>` passed a random choice
    - A `<Result>` passed both picks, shows winner