Logistics

- Projects going well? More Office Hours?
- Everyone have create-react-app working?
- HackUVA participants, whatcha end up building?
- React Hackathon?

React

An Evening With Michael Crawford

Classes

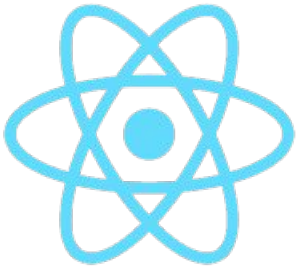
- Similar idea to classes in Java or C++
- Can be thought of as a “blueprint” that specifies attributes (adjectives) and methods (verbs)
- Contains “constructor” ➔ method to create specific instance
- The syntax we’ll use to create a React component

```
class SkinnedMesh extends THREE.Mesh {
  constructor(geometry, materials) {
    super(geometry, materials);

    this.idMatrix = SkinnedMesh.defaultMatrix();
    this.bones = [];
    this.boneMatrices = [];
    //...
  }
  update(camera) {
    //...
    super.update();
  }
  get boneCount() {
    return this.bones.length;
  }
  set matrixType(matrixType) {
    this.idMatrix = SkinnedMesh[matrixType]();
  }
  static defaultMatrix() {
    return new THREE.Matrix4();
  }
}
```

What IS React?

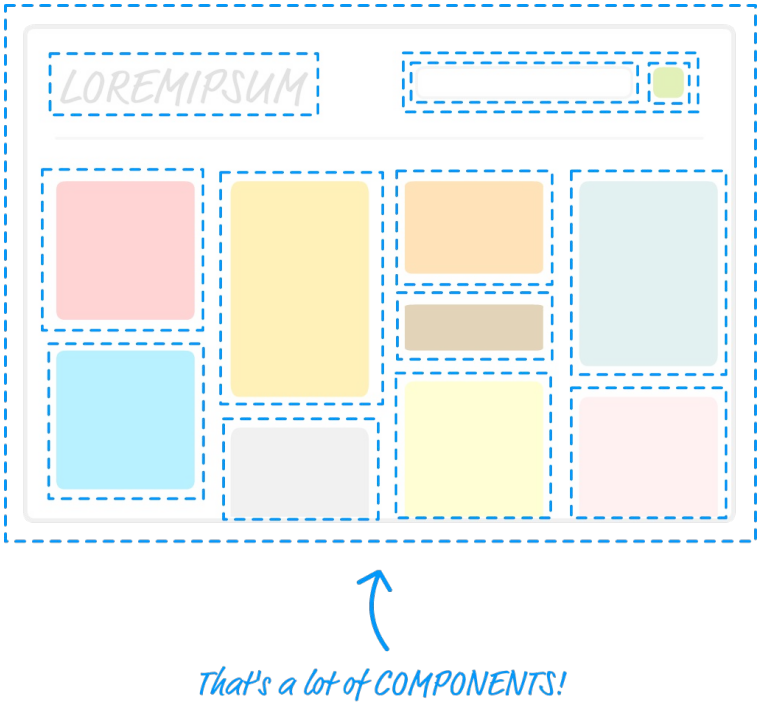
- JavaScript Framework for front-end development
- Can be thought of as collapsing our HTML, CSS, and JavaScript all into one
- Allows us to think in terms of components



React

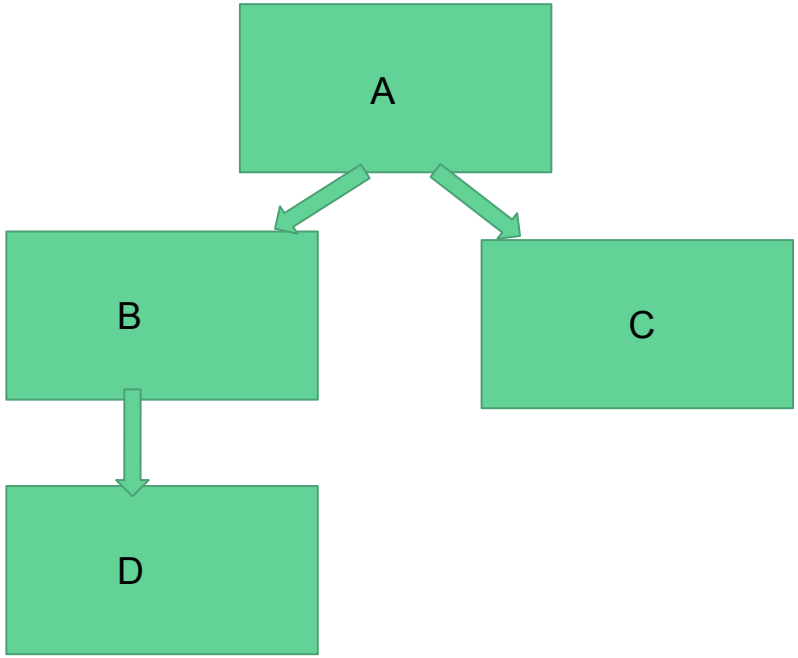
Component-Based Architecture

- Each individual part of webpage conceptualized as a “component”
- If different things do similar (or exact same) things, we can write a single “generic” component to cover them



Unidirectional Dataflow

- The way components are laid out can be thought of as a tree
  - A subtree would be rendered inside the parent
- Relevant data flows down through the tree
- Managed in the state of a shared parent component
- Let’s draw a picture!



Virtual DOM and Single-Page Web Apps

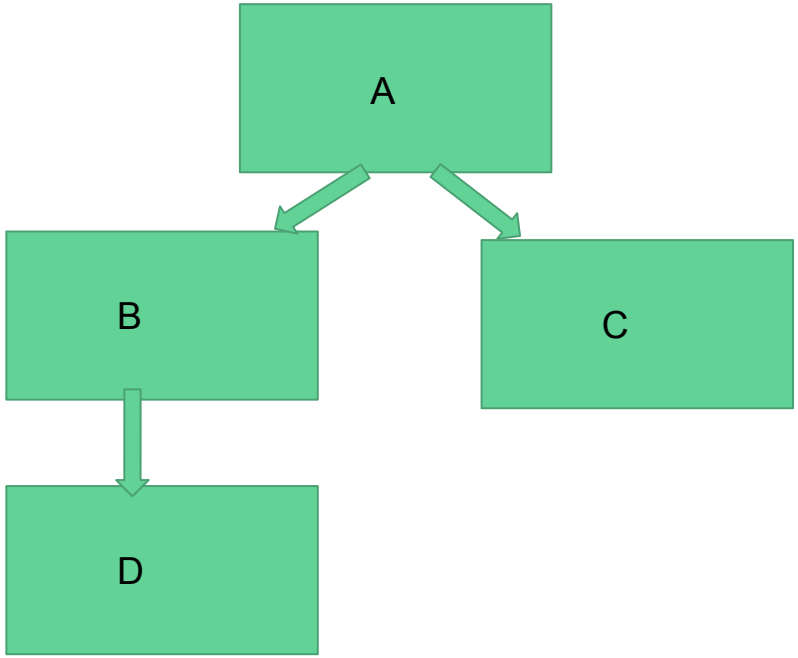
- Instead of reloading web pages, React updates the Virtual DOM accordingly
- Speedier and more responsive web apps

What goes into a React Component?

- State versus Props
- Constructor if we’re managing state
- Relevant methods
- render() method

State versus Props

- State is data that this component is responsible for managing and updating
  - Initialized with a constructor method, stored as a JavaScript Object
  - Updated using this.setState(newState), where newState is the new Object
  - WARNING: this method is asynchronous!
  - **Changing the state rerenders the component and its children**
- Props is data passed down from the parent
  - Think of as parameters for a function



Constructor

- Generally always use the same formula:
  - Method called constructor(props)
  - First call super(props)
  - Set the initial state with this.state = { }

```
export default class ApproveOrDenyUserList extends Component {
  constructor(props) {
    super(props);
    this.state = {
      users: [],
      status: "Loading..."
    };
  }
```

Other Methods

- Can write any old method you'll need for this component
- For example, updating the state to increase a count, etc.
- How you make your components interactive!

Render Method

- This method tells React what you actually want displayed
- Uses JSX (kinda like merging your HTML with JavaScript)
- You can render standard HTML tags like <p>, <input>, <h1>, etc.
- Can also render other components!
- Let’s write a little example on the board

Render Method and JSX Syntax

- Use .map() if you need an array of components
- Then in your return statement, write up everything you'll need using HTML style syntax
- Props (and any other JavaScript code/variables your JSX needs to reference) are passed in using curly braces { }

```
render() {
  const roles = this.props.roles.map(role => {
    return (
      <Select.Option key={role} value={role}>
        {role}
      </Select.Option>
    );
  });
  return (
    <div>
      <Select
        onChange={val => this.onChangeValue(val)}
        style={{ width: "150px" }}
        placeholder="Select Role"
      >
        {roles}
      </Select>
      <Button onClick={() => this.onClickButton()}> Save </Button>
    </div>
  );
}
```

Let's Write a Component Together!!