ReactJS — Components

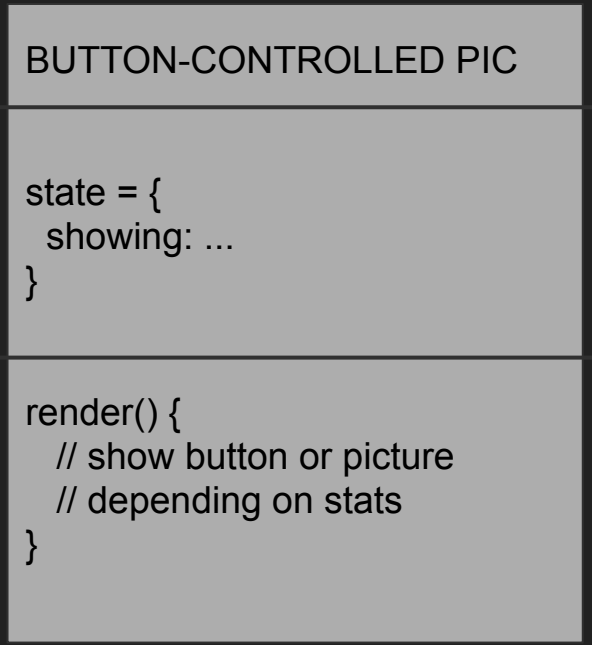
Web Dev, Spring 2021

ReactJS components

Component = class whose objects have:

- a render method that returns the HTML + CSS of the component
- some component state
- every time the component state changes, ReactJS re-renders the component (recreates the HTML)

Can attach a component anywhere in an HTML document



A React Component with State

```
class ButtonPicture extends React.Component {
  constructor(props) {
    super(props)
    this.state = {showing: false}
  }
  render() {
    const name = this.props.name
    const url = this.props.url
    const click = () => { this.setState({showing: true}) }
    if (this.state.showing) {
      return ( <div className="column">
                 <div>{name}</div>
                 <img src={url} />
               </div>  )
    } else {
      return <button onClick={click}>Show picture</button>
    }
}
```

The Basic Recipe

(1) Figure out initial arguments to the component

(2) Figure out state you need to maintain

(3) Write a render method returning HTML code based on state (and arguments)

(4) Add event handlers to change the state when needed

Example: Sortable Table

(1) Figure out initial arguments to the component

Rows to show in the table, list of headers

(2) Figure out state you need to maintain

Sorted rows, sorting column

(3) Write a render method returning HTML code based on state (and arguments)

Create a table with the sorted rows and highlight sorting column

(4) Add event handlers to change the state when needed

Click on a header cell → sort the rows according to header that was clicked

Example: Sortable Table

```
class SortableTable extends React.Component {
  constructor(props) {
    super(props)
    this.state = {sortedRows: props.rows, sorting: -1}
                                              // < 0 → not sorted
  }
  sortRows(col) {
    const newRows = this.state.sortedRows.slice()
    newRows.sort((a, b) => a[col].toString().localeCompare(b[col].toString()))
    return newRows
  }
  render() {
    ...
  }
}
```

Trick — Rendering an array of elements

If `arr` is an array of JSX expressions, then using

```
{ arr }
```

in a JSX expression will inject all elements of the array into the JSX as siblings

Examples:

```
{ [<div>x</div>, <div>y</div>, <div>z</div>] }
                          → <div>x</div> <div>y</div> <div>z</div>
{ ['x', 'y', 'z'].map(item => <div>{ item}</div>) }
                          → <div>x</div> <div>y</div> <div>z</div>
```

Example: Sortable Table (continued)

```
render() {


  const rows = this.state.sortedRows
  const headers = this.props.headers
  return ( <table>
            <thead>
              <tr>{ headers.map(h => <th>{ h }</th>) }
              </tr>
            </thead>
            <tbody>
              { rows.map(r => <tr>{ r.map(item => <td>{ item }</td>) }</tr>) }
            </tbody>
           </table> )
}
```

Example: Sortable Table (continued)

```
render() {
  const handleClick = (col) => () => {
    this.setState({sortedRows: this.sortRows(col), sorting: col})
  }
  const rows = this.state.sortedRows
  const headers = this.props.headers
  return ( <table>
            <thead>
              <tr>{ headers.map((h, i) => <th onClick={handleClick(i)}>{ h }</th>) }
              </tr>
            </thead>
            <tbody>
              { rows.map(r => <tr>{ r.map(item => <td>{ item }</td>) }</tr>) }
            </tbody>
           </table> )
}
```

Demo

Sortable table, filterable list

Using Components in JSX expressions

You can use a component within a JSX expression

```
<Component x="..." y="..." />

<Component x="..." y="..."> ... </Component>
```

The attributes will be passed as initial arguments (props) of the component

This means a component may renders using child components

- Whenever a component renders, child components are re-created
- If the props passed to a child depend on component's state,
  you can affect how that child renders

Demo

Combining components