ReactJS — Introduction

Web Dev, Spring 2021

Revisiting the frontend

Raw HTML + Javascript:

- Code management and reuse is complex
  - MVC pattern is complex to maintain
  - HTML and Javascript are separate and need to be kept in sync

Frameworks for frontend development

- Angular
- Vue.js
- ReactJS

They all provide infrastructure to manage complexity (management + reuse)

Frontend frameworks

Usually based on components

- a component is an encapsulation of HTML elements with some logic associated with them
- may use subcomponents

Main goal of a component is reusability

- you should be able to use or reuse a component in various places easily

ReactJS

- open-source
- developed by Facebook
- focused on user interfaces components
- geared toward single-page apps and mobile apps

Needs some help to handle:

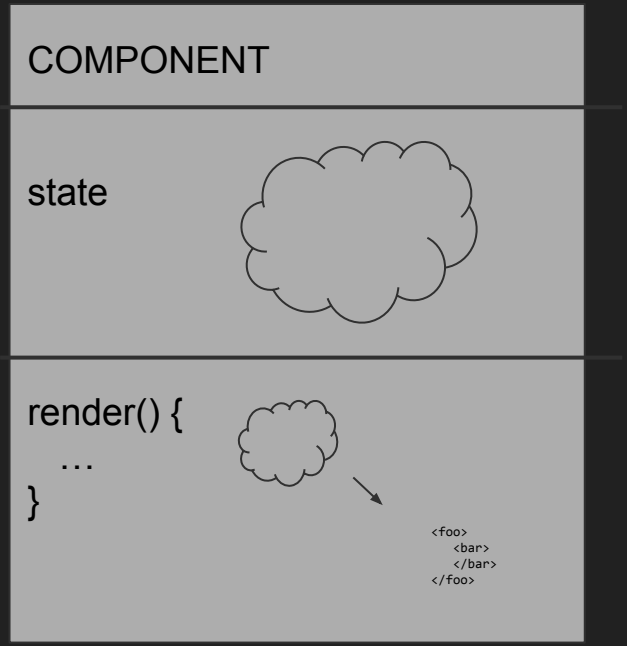
- global state management
- routing
- other frontend tasks...

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

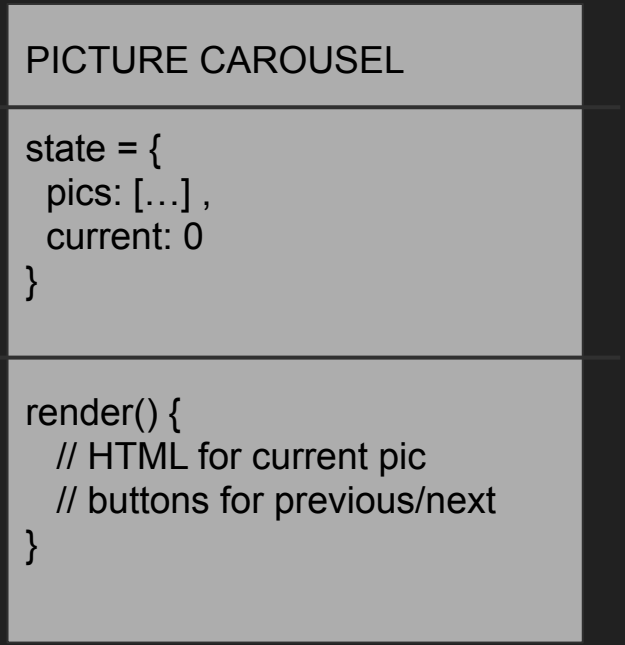


ReactJS components

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 Simple React Component

```
class Picture extends React.Component {
  render() {
    const name = this.props.name
    const url = this.props.url
    return ( <div className="column">
               <div>{name}</div>
               <img src={url} />
             </div>  )
   }
}
```

A Simple React Component

```
class Picture extends React.Component {
  render() {
    const name = this.props.name
    const url = this.props.url
    return ( <div className="column">
               <div>{name}</div>
               <img src={url} />
             </div>  )
   }
}
```

(Assumes React is in the global environment)

A component extends React.Component

Must implement a render() method

A Simple React Component

```
class Picture extends React.Component {
  render() {
    const name = this.props.name
    const url = this.props.url
    return ( <div className="column">
               <div>{name}</div>
               <img src={url} />
             </div>  )
   }
}
```

A component can be parameterized

- like function parameters

Parameters are called `props`, and they are available in the this.props field (automatically populated by React.Component)

Picture will be instantiated with {name: …, url: …}

A Simple React Component

```
class Picture extends React.Component {
  render() {
    const name = this.props.name
    const url = this.props.url
    return ( <div className="column">
               <div>{name}</div>
               <img src={url} />
             </div>  )
   }
}
```

The render method must return an HTML element

- either via React.createElement
- or with JSX

JSX is a special notation for writing HTML as a "value" in JS

Need some special setup for handling JSX

A Simple React Component

```
class Picture extends React.Component {
  render() {
    const name = this.props.name
    const url = this.props.url
    return ( <div className="column">
               <div>{name}</div>
               <img src={url} />
             </div>  )
   }
}
```

JSX can inject Javascript values into the resulting HTML code

A Less Simple React Component

```
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>
    }
}
```

A Less Simple React Component

```
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>
    }
}
```

Constructor to set up the initial state

Component state must be in this.state

A Less Simple React Component

```
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>
    }
}
```

State should be an object

Can use props (the parameters to the component) to set up the initial state

A Less Simple React Component

```
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>
    }
}
```

Can use the state in the render() method

Here, show the image or a button depending on this.state.showing

A Less Simple React Component

```
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>
    }
}
```

Can associate a function with the `click` event with `onClick`

That function here updates the state via

this.setState({...})

React will see the state change and re-render the component!