CS390 WEB APPLICATION DEVELOPMENT

NISARG KOLHE

Announcements

- Lab 2 has been extended to be due on **Friday, September 20th** at midnight.
- If you were told to change your project plan, submit your **updated** project proposal on Blackboard.
- You are provided with **2** excused absences.

Let’s talk about packages.

Using an external library

Just link the external JS file.

Nothing can go wrong with that, right?

```
…
<script src="moment.js"></script>
<script src="script.js"></script>
…
```

Problems with just linking the external JS file

- Can't keep **track** of new versions if the library is constantly updated.
- If linking directly from source, changes are **unpredictable**.
    - A new version could be released anytime changing the API and breaking your code.
    - Or the website could just go down and take your app down with it.

Problems with just linking the external JS file

- The library is loaded into a **global** variable
- The browser has to make one **additional** request to load your website
    - Lesser the requests, faster the website!

Introducing, NPM.

Node

Package

Manager

World's largest software repository of re-usable packages built for JavaScript.

Importing packages with NPM

Step 0 : Make sure your project is initialized with NPM (this creates **package.json** file)

```
$ npm init
```

Step 1 : Install the package (this downloads to package to **node_modules** and updates **package.json** file)

```
$ npm install --save moment
```

Step 2 : Use the package in your code!

```
import moment from ‘moment’;
```

Importing packages with NPM

- This won’t work in browser, since there’s no such thing as **require** in vanilla JavaScript.
- Need to use a **module bundler** to scan JS files
  - A bundler looks for require keyword and replaces it with the **entire** content of the file
  - In the end, we’re left with a single JS file containing all the code we need

Node.JS?

We'll cover it later in the course

Module Bundler?

React comes with one! No time to discuss others.

React

Products built with React

airbnb

asana

React

…is a JavaScript library for building user interfaces.

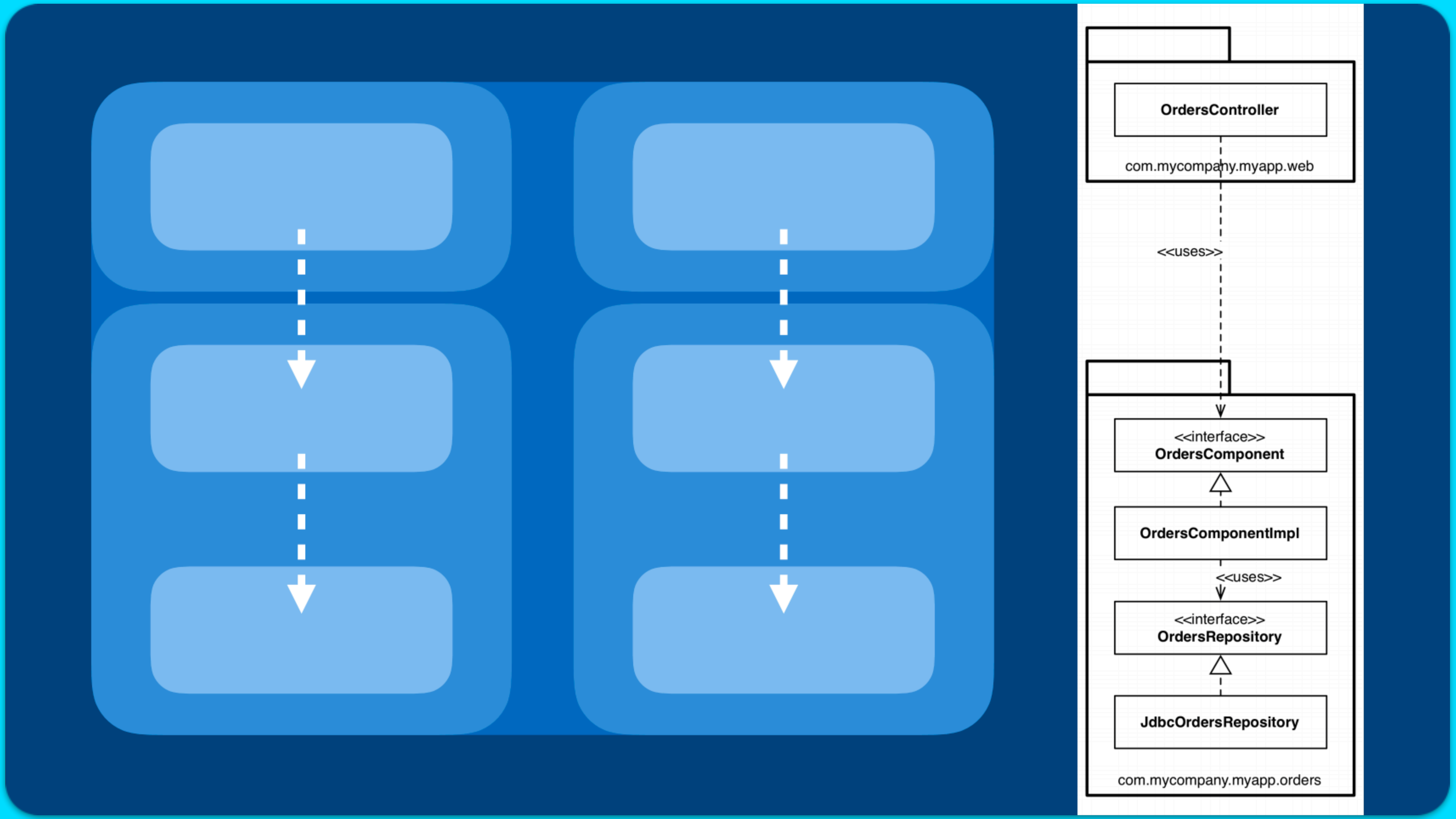
Built by Facebook for making web development more modular.

React is NOT

- A design framework
    - Doesn’t automatically make your website look better
- A backend framework
    - Lives on the frontend, though you can generate static files to serve from a backend
- A framework
    - It’s a library. Needs other libraries like Redux to behave like a complete framework

React

Uses **component** based architecture.



Using React (the easy way)

Step 1: Generate boilerplate code

```
$ npx create-react-app my-app
```

Step 2: Run the local server

```
$ npm start
```

Anatomy of a stateful React component

ES5 Class extending React.Component

Simple constructor

Component's state

HTML to be rendered by this component

```
class C extends React.Component {
   constructor {
      this.state = {
         msg = ‘Hello World!'
      }
   }

   render() {
      return
         <h1>
             {this.state.msg}
         </h1>
   }
}
```

What is **State**?

Think of **state** as data you want your component to

“**remember**”.

Properties of **State**

- **Immutable**
  - React depends on checking for changes in state to know when to re-render the components
  - Can’t just change state by reassigning the values
- Can’t be updated inside **render()**, needs to have definite value before rendering.

How not to update State

```
class C extends React.Component {
   constructor {
      this.state = {
         msg = ‘Hello World!'
      }
   }

   click() {
      this.state.msg = ‘Bye!’;
   }

   render() {
     <>
      <h1>{this.state.msg}</h1>
      <button onClick={() => click()}/>
     </>
   }
}
```

How to update **State**

```
class C extends React.Component {
   constructor {
      this.state = {
         msg = ‘Hello World!'
      }
   }

   click() {
      this.setState{msg: ‘Bye!’};
   }

   render() {
     <>
      <h1>{this.state.msg}</h1>
      <button onClick={() => click()}/>
     </>
   }
}
```

render()

- Returns a JSX object to be rendered by the component
  - JSX converts HTML looking code into JavaScript

```
render() {
    <div>
      …
    </div>
}
```

render()

To render more than one element, wrap them in **React.Fragment**.

```
render() {
  <React.Fragment>
   <div></div>
   <div></div>
  </React.Fragment>
}
```

=

```
render() {
  <>
   <div></div>
   <div></div>
  </>
}
```

props

You can pass properties to components!
Properties are written just like attributes in HTML.

```
class Parent extends React.Component
{
   render() {
     <Child msg={‘Hi!’}/>
   }
}
```

```
class Child extends React.Component
{
   render() {
     <h1>{this.props.msg}</h1>
   }
}
```

Resources

- Modern JavaScript Explained For Dinosaurs
  - https://medium.com/the-node-js-collection/modern-javascript-explained-for-dinosaurs-f695e9747b70
- How JavaScript bundlers work
  - https://medium.com/@gimenete/how-javascript-bundlers-work-1fc0d0caf2da
- React Docs
  - https://reactjs.org/docs/hello-world.html