ECE 448/528 Application Software Design

Lecture 21. The React JavaScript Library

Spring 2025

Won-Jae Yi, Ph.D.

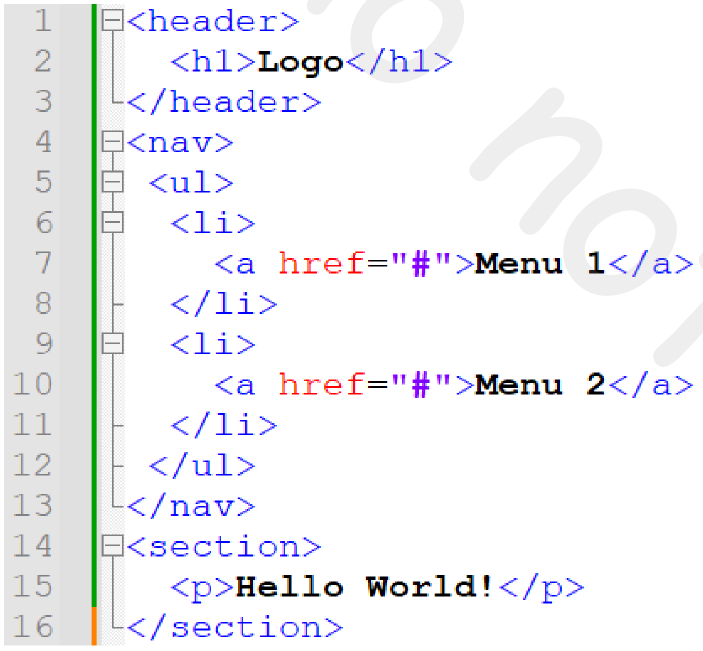
Department of Electrical and Computer Engineering
Illinois Institute of Technology

The React JavaScript Library

React

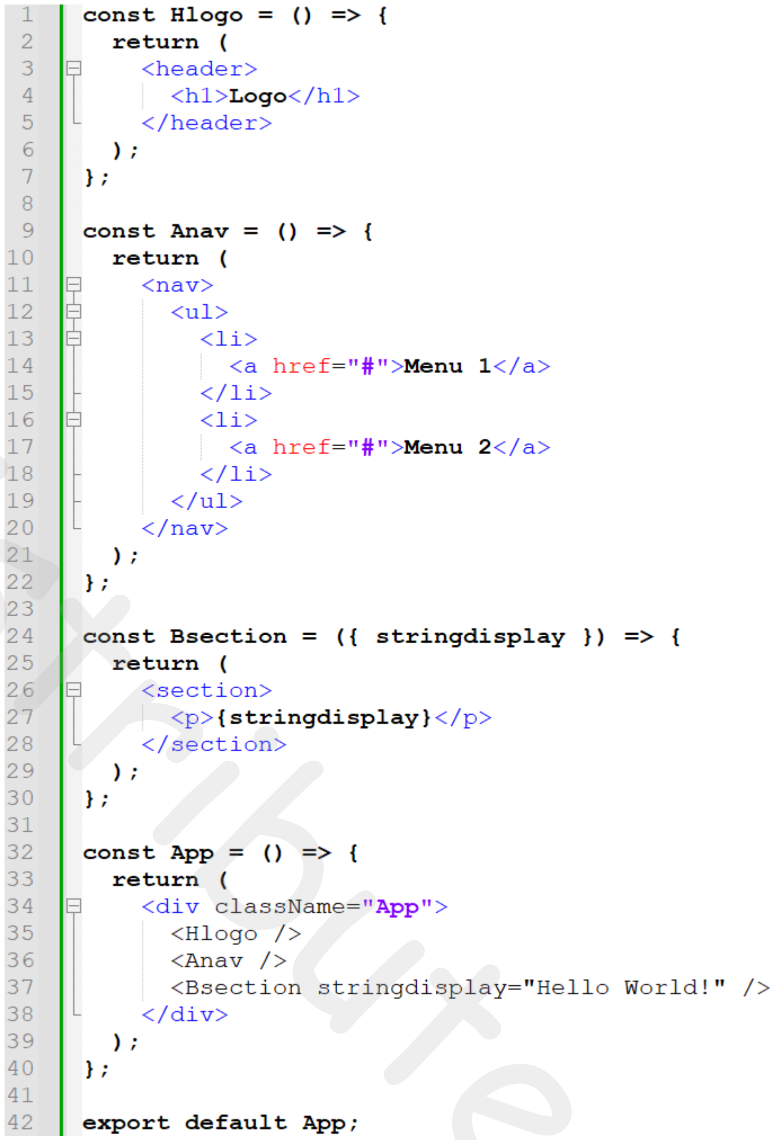
- A JavaScript library for building (complex) user interfaces, since 2013.
  - Created for web applications by Facebook (Meta) but now applies to mobile and other applications that require UI.
  - Single Page Application (SPA); downloads massive amount of JavaScript from the server, then renders only necessary views dynamically with minimal information based on the server response
  - A popular choice to build web applications now.
    - Other popular choices: AngularJS (Google), Vue.js
- Precautions
  - React is complex and flexible due to the nature of UI design.
  - While there are many official and unofficial tutorials and guides, you may quickly feel lost because of the many other new tools and new associated libraries that are introduced at the same time
- Our approach in this course will focus on the core concepts of MVC to allow you to learn React and future UI libraries quickly, minimizing the need to use additional tools and libraries. It is not necessarily the best practice, but it enables you to explore React further.

React - Example



```
<header>
  <h1>Logo</h1>
</header>
<nav>
 <ul>
  <li>
    <a href="#">Menu 1</a>
  </li>
  <li>
    <a href="#">Menu 2</a>
  </li>
 </ul>
</nav>
<section>
  <p>Hello World!</p>
</section>
```

Without using React



```
const Hlogo = () => {
  return (
    <header>
      <h1>Logo</h1>
    </header>
  );
};

const Anav = () => {
  return (
    <nav>
      <ul>
        <li>
          <a href="#">Menu 1</a>
        </li>
        <li>
          <a href="#">Menu 2</a>
        </li>
      </ul>
    </nav>
  );
};

const Bsection = ({ stringdisplay }) => {
  return (
    <section>
      <p>{stringdisplay}</p>
    </section>
  );
};

const App = () => {
  return (
    <div className="App">
      <Hlogo />
      <Anav />
      <Bsection stringdisplay="Hello World!" />
    </div>
  );
};

export default App;
```

With using React

Language Support for React

- React can be written in a few different ways.
- We'll write React using JavaScript with JSX extension.
    - JSX (JavaScript XML) is a syntax extension to JavaScript, which allows writing HTML in React.
    - When necessary files are included, our code can run directly in the browser – very helpful for learning React quickly.
    - A disadvantage is that resources may be wasted as not all parts from those included files are necessary.
- Alternatively, one may set up a full development environment that compiles your React code into plain JavaScript.
    - Use better language like TypeScript (TS) (at the professional level).
        - Uses same syntax as JavaScript but needs to declare 'type' to avoid runtime errors in JavaScript
    - Providing additional tools to support modularity and troubleshooting.
    - Minimizing the bytes transferred when loading the web page.
    - You may take this approach if you prefer to, though we won't be able to cover it in the lectures.

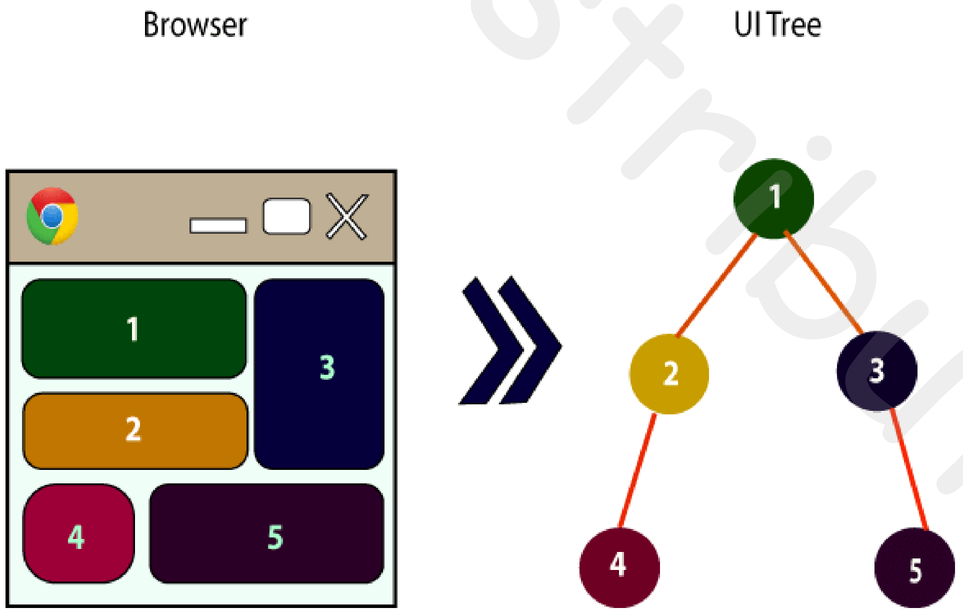
UI Design with React

UI Design with React

- Let's continue our UI design for the groups but make use of React.
  - First step done: UI mockup
- User story to be addressed: As an end-user, I want to list groups and members belonging to and not belonging to group(s) so that I can see group/member relations for a specific member.
- Use React to generate the table.
  - As the View in MVC.
  - With sample data
  - We will also design and implement the Model. We're not ready yet to communicate with the RESTful server, so ignore event handlers for buttons and checkboxes.

React Components

- In React, the web UI on a single web page is organized into React components.
  - Recall that the HTML DOM (Document Object Model) defines the structure of the web UI.
  - Each React component is responsible for a sub-tree in the HTML DOM.
  - Like the HTML DOM, React components are composed into a tree-like hierarchy.



React Components

- **State** in React
  - A built-in plain JavaScript object used to contain data/information about the component
  - Used to store the data of the components that need to be rendered to the view
- **Props** in React
  - A built-in plain JavaScript object used to pass data and event handlers to the children's components
- Two types of component designs in React
  - Stateless/functional component, to work only as Views. (no state, no props)
  - Stateful component, to provide full support of MVC pattern.
- We will…
  - Use stateless components whenever possible, which are only Views.
  - Stateful components should focus on the Model and Controller and avoid providing View functionality.

The HTML Container

```
<head>
  . . .
  <!-- Members App -->
  <script type="text/babel" src="web/members_app.js"></script>
  <script type="text/babel" src="web/members.js"></script>
  <script type="text/babel" src="web/members_table.js"></script>
  . .
</head>
```

- In `public/members.html` under branch `lec21-react`
- We implement the application in React with code in the web directory.
  - Note that the "text/babel" attributes allow the browser to interpret those scripts as JSX code for React.
  - `members_app.js`: top level Controller for the application
  - `members.js`: Controller and Model for the member's table.
  - `members_table.js`: Views for the member's table.

The HTML Container

```
<body>
  <div id="app_container" />
  <script type="text/babel">
    $(document).ready(() => {
      ReactDOM.render(
        <MembersApp />,
        document.getElementById("app_container"));
    });
  </script>
</body>
```

- Usual routine to start a React web application.
  - Provide a div for the View of the web application.
  - When the HTML DOM is fully loaded, create the top-level application component and associate it with the div.
- The script with "text/babel" attributes is in JSX
  - A mixture of JavaScript code and HTML-like tags.

The HTML Container

```
<div id="app_container" />
<script type="text/babel">
  $(document).ready(() => {
    ReactDOM.render(
      <MembersApp />,
      document.getElementById("app_container"));
  });
</script>
```

- The event that the HTML DOM is fully loaded is captured by the function $(document).ready from the JQuery library.
  - The lambda function works as an event handler.
- The top-level application component is created by <MembersApp />.
  - Conceptually, this special JSX syntax is similar to that of new in Java.
  - <MembersApp /> is an object of the type MembersApp.
- ReactDOM.render associates the MembersApp component with the div.
  - document.getElementById("app_container") locates the div with in the HTML DOM.

The Top-Level Application Component

```
//  public/web/members_app.js
class MembersApp extends React.Component {
   constructor(props) {super(props);}
   . . .
}
window.MembersApp = MembersApp;
```

- Extend React.Component to create stateful component.
  - The top-level component is a Controller.
  - Use recent JavaScript updates of the familiar OOP syntax– however, there are numerous conditions, so we need to be very careful.
- The scripts are in JSX where modularity is enforced.
  - Symbols like names of functions and classes defined in individual files are local.
  - e.g., the MembersApp class defined here is not available in other files like members.html.
  - We can make symbols globally available by creating entries in the global window variable.

The Constructor

```
//  public/web/members_app.js
class MembersApp extends React.Component {
   constructor(props) {super(props);}
   ...
}
window.MembersApp = MembersApp;
```

- constructor is called when the component is created.
- Constructors of React components only take one parameter props.
- We will discuss the props parameter later.

The render() Function

```
class MembersApp extends React.Component {
  ...
  render() {
    return (
      <div className="container">
        <div className="row">
          <div className="col-sm-12">
            <h2>Manage Groups and Members</h2>
            <Members />
          </div>
        </div>
      </div>);
  }
}
```

- The member function render() creates the view by defining part of the HTML DOM.
  - Try to compare this piece of code with that from the UI mockup.
- The HTML-like tags may have different attribute names than HTML, e.g., use className for class.

A Hierarchy of React Components

```
render() {
  return (
    <div className="container">
      . . .
          <Members />
      . . .
    </div>);
}
```

- The render() function can create a hierarchy of React components.
  - e.g., <Members /> – recall that the syntax <Members /> is used to create a React component of the type Members.
- Later, we will learn how to pass models in render() among React components.
  - Our application is simple, so there is no need to define models at the top level.
  - But more complex applications, like the one coming with the project, may define models at the top-level and share them between lower-level React components.

Defining Models in JavaScript

The Members Model

```
// public/web/members.js
function create_members_model(groups) {
  var all_members = new Set(); // all unique member names
  var group_names = [];
  var group_members = new Map(); // group_name to set of group members
  ... // populate variables from groups
  var member_names = Array.from(all_members);
  group_names.sort();
  member_names.sort();
  ...
```

- We need a “members” model to support search for members in groups.
- The model should be created from the groups response from the RESTful backend.
- In addition to JSON arrays and objects, Set and Map are useful data structures you may use.

OOP or Not OOP

```
// public/web/members.js
function create_members_model(groups) {
  var all_members = new Set(); // all unique member names
  var group_names = [];
  var group_members = new Map(); // group_name to set of group members
  ... // populate variables from groups
  var member_names = Array.from(all_members);
  group_names.sort();
  member_names.sort();
  ...
```

- Like our RESTful models in Java, you may attempt to define models using the OOP syntax recently introduced to JavaScript.
- Unfortunately, that is NOT a good idea because that OOP syntax in JavaScript is somewhat broken.
  - A lot of boilerplate (sections of repetitive or standard code) code is required to make the OOP syntax work.
  - The React library takes some care of that, so we can use the OOP syntax to define React components.
- Let’s create models in JavaScript in the traditional JavaScript

this and that

```
function create_members_model(groups) {
  . . .
  var that = {}
  that.get_group_names = () => group_names;
  that.get_member_names = () => member_names;
  that.is_member_in_group = (member_name, group_name) =>
    !group_members.has(group_name)? false:
      group_members.get(group_name).has(member_name);
  return that;
}
```

- JavaScript objects with methods and private data can be created on the fly using lambda functions and closures.
- Use that to refer to the object since this is a JavaScript keyword.
  - Be careful; this in JavaScript is very different than this in other OOP languages.
- What's inside the object that?
  - Three members: get_group_names, get_member_names, is_member_in_group. All are functions.

Closure

```
function create_members_model(groups) {
  ...
  var that = {}
  that.get_group_names = () => group_names;
  that.get_member_names = () => member_names;
  that.is_member_in_group = (member_name, group_name) =>
    !group_members.has(group_name)? false:
      group_members.get(group_name).has(member_name);
  return that;
}
```

- Each of these three functions manipulates local variables of the create_members_model function.
  - The object that contains no data at all.
  - The referred local variables are kept in a <u>closure</u> so they can be used even after create_members_model returns.
  - Multiple calls to create_members_model will generate different sets of local variables, and thus different closures and different objects.
- The use of closures ensures no one can circumvent the methods to modify the "private" data.