KINGSLAND
UNIVERSITY

Introduction to React.js

What is React, JSX, Overview and Syntax

Table of Contents

1. React Overview
2. Installation
3. JSX Syntax
4. Composition

React Overview

History and Philosophy

What is React.js?

- **React** is a JavaScript library for building **user interfaces** (UI)
- Focused on creating **reusable components**
- Developed by **Facebook**

```
const HelloMessage = (props) => (
  <div>Hello {props.name}</div>
);
ReactDOM.render(
  <HelloMessage name="Maria" />,
  document.getElementById('root')
);
```

Features

- Open-source
- Declarative
  - Design **simple** views for each **state** in your app
  - Easier to **debug**
- Component-Based
  - Encapsulated **components** that manage their **own** state
  - Keep **state** out of the **DOM**

Features

- ✓ Isomorphic
  - ✓ JavaScript that runs on **both** client & server
  - ✓ Better user experience
- ✓ Native support
  - ✓ Compose rich **mobile** UI in **Android, iOS**

Advantages

- Easy to learn
- Fast **performance**
- Use all **ES6** features
  - **Promises**, **Classes** and **Modules**
- Compatible with other **libraries**
- Great **error reporting**

React Installation

Packages, Setup, Structure

Create React App

- ✓ Less to learn - **instant reloads** help you focus on development
- ✓ Only one dependency - no complicated version mismatches
- ✓ No Lock-In - under the hood **Webpack**, **Babel**, **ESLint**
- ✓ Install the React app creator (one-time global install)

Install and Run the React App Creator

- Run the React app creator

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

- Starts your React app from the command line

```
cd my-app
```

```
npm start
```

- Browse your app from http://localhost:3000

Finding Information

- ✓ Visit the **official website**

  https://reactjs.org/

- ✓ Documentation

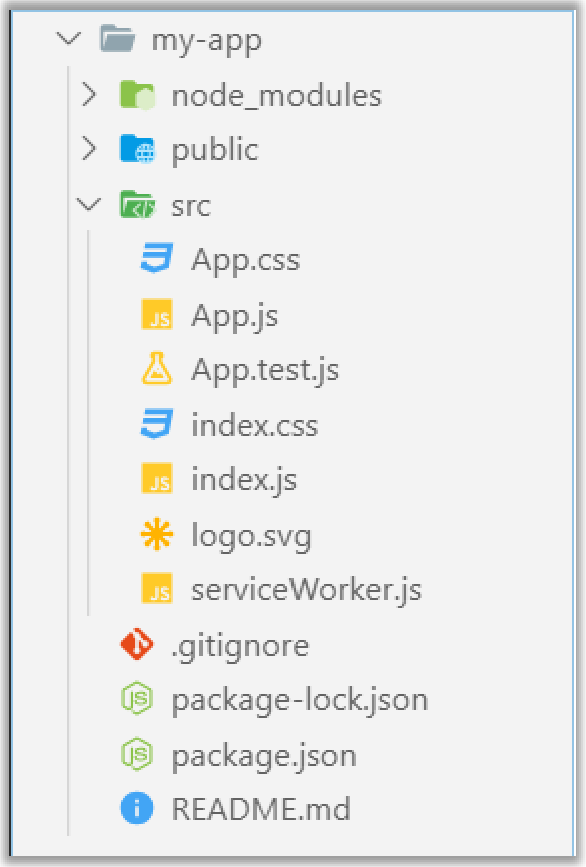
  https://reactjs.org/docs/installation.html

- ✓ Online sandbox

  https://codesandbox.io/

React App Structure

- **`package.json`** - project configuration
  - Module name, dependencies, build actions
- **`index.html`**
  - App main HTML file
- **`index.js`**
  - App main JS file (startup script)
- **`App.js`**, **`App.css`**, **`App.test.js`**
  - React component "**App**"



```
my-app
  node_modules
  public
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    serviceWorker.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

Live Demo

JSX Syntax

Overview, Syntax, Advantages

JSX Overview

- **JSX** is React's JavaScript **superset language**
    - Has all of JavaScript's **features** and more
- Unique approach to **mixing HTML and JS**
- Compiles to **plain JavaScript**

```
<div className="red">Children Text</div>
```

```
React.createElement("div",
  { className: "red" },
  "Children Text"
);
```

JSX Syntax

```
<div className="red">Children Text</div>
```

```
<MyCounter count={3 + 5} />
```

Expression

```
let gameScores = {
  player1: 2,
  player2: 5
};
<DashboardUnit index="2" onClick={() => {}}>
  <h1>Scores</h1>
  <Scoreboard className="results" scores={gameScores} />
</DashboardUnit>
```

Custom component

Pass variable as prop

JSX Rules and Principles

- Standard elements use lowercase names
  - **`div`**, **`span`**, **`form`**, **`input`**, …
- Custom components **always** use Pascal case
  - **`MyCustomComponent`**, **`Greeting`**, **`ScoreBoard`**, …
- Component name cannot be an expression
  - Use a **variable** instead
- There must be a **root element**
- More info at: https://reactjs.org/docs/jsx-in-depth.html

Compilation

- JSX **compiles** to function calls

```
<div className="red">Children Text</div>
```

Element type (HTML tag name)

Properties object

List of children

```
React.createElement("div",
                    { className: "red" },
                    "Children Text" [, …]);
```

Live Demo

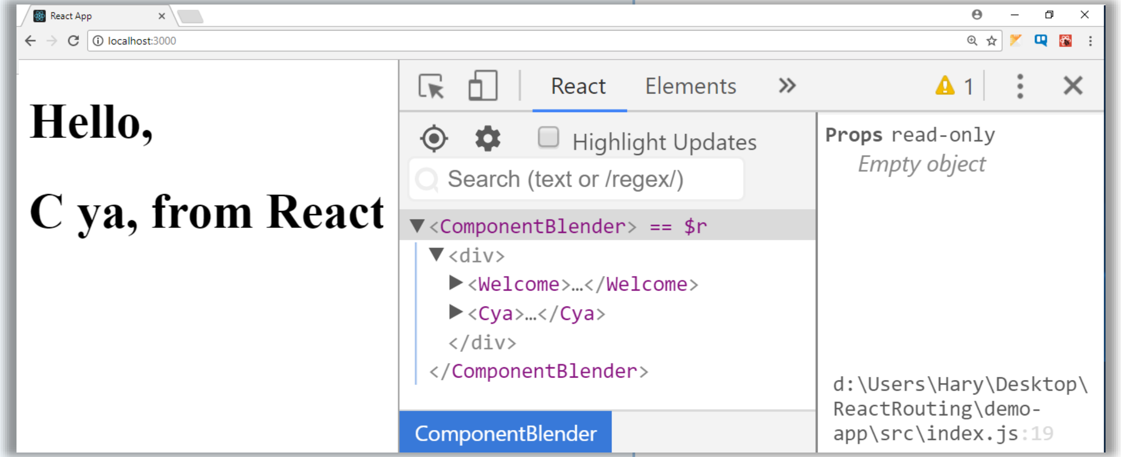
Composition

Definition and Advantages

Composition

- React components can be **nested**, like DOM elements

```
function Welcome() {
  return <h1>Hello, from React</h1>;
}
function Cya() {
  return <h1>C ya, from React</h1>;
}
function ComponentBlender() {
  return (
    <div>
      <Welcome />
      <Cya />
    </div>
  );
}
```



Hello,

C ya, from React

```
▼<ComponentBlender> == $r
  ▼<div>
    ▶<Welcome>…</Welcome>
    ▶<Cya>…</Cya>
   </div>
 </ComponentBlender>
```

Props read-only
Empty object

d:\Users\Hary\Desktop\ReactRouting\demo-app\src\index.js:19

```
ReactDOM.render(<ComponentBlender />,
    document.getElementById('root'));
```

Component Syntax

- Names always start with **uppercase**
- Tags must be **closed**
  - If there are no children - use **self-closing tags**
- **Information** is passed via **props**

```
<Dropdown> A dropdown list <UserHead name='homeHeader' />
  <Menu>
    <MenuItem>Do Something</MenuItem>
    <MenuItem>Do Something Fun!</MenuItem>
    <MenuItem>Do Something Else</MenuItem>
  </Menu>
</Dropdown>
```

Advantages

- Encapsulate logic
- **Separate** your code
  - Easier to **maintain** and **debug**
  - Allows **reusability**
- Components are neat

Live Demo

Summary

- **React** is a JavaScript library for building user interfaces
- **React** uses all **ES6** features
- **JSX** is React's JavaScript **superset**
- React **components** can be nested (composition)

Questions?

License

- This course (slides, examples, demos, exercises, homework, documents, vide and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © Kingsland University – https://kingslanduniversity.com

KINGSLAND UNIVERSITY

THANK YOU