Web Programming

CS-406

Lecture # 06

React

Framework and Library

- The key difference between JavaScript libraries and frameworks is that:
- libraries consist of functions that an application can call to perform a task.
- While a framework defines how a developer designs an application.
- In other words, the framework calls on the application code, rather than the other way around.
- When using a library, you are in charge of the flow of the application. You are choosing when and where to call the library.
- When you using a framework, the framework is in charge of the flow. It provides some places for you to plug in your code, but it calls the code you plugged in as needed.

React

- React is a JavaScript library for building user interfaces created by Facebook.
- React is used to build single page applications.
- React allows us to create reusable UI components.
- Also called React.js or ReactJS
- React Native is used for development of Mobile applications
- React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.
- Often referred to as frontend framework
- Used for creating fast and interactive interfaces.

Working

- **React creates a VIRTUAL DOM in memory.**

Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

- **React only changes what needs to be changed!**

React finds out what changes have been made, and changes only what needs to be changed.

- This process is called **reconciliation**. This allows the programmer to write code as if the entire page is rendered on each change, while the React libraries only render subcomponents that actually change.
- This selective rendering provides a major performance boost. It saves the effort of recalculating the CSS style, layout for the page and rendering for the entire page

History

- Current version of React.JS is V16.8.6 (March 2019).
- Initial Release to the Public (V0.3.0) was in July 2013.
- React.JS was first used in 2011 for Facebook's Newsfeed feature.
- Facebook Software Engineer, Jordan Walke, created it.
- The create-react-app version 2.0 package was released in October 2018.
- Create-react-app version 2.0 supports Babel 7, webpack 4, and Jest23.

React and ES6

React uses ES6, and you should be familiar with some of the new features like:

- Classes
- Arrow Functions
- Variables (let, const, var)

Classes

```
class Car {
  constructor(name) {
    this.brand = name;
  }

  present() {
    return 'I have a ' + this.brand;
  }
}

class Model extends Car {
  constructor(name, mod) {
    super(name);
    this.model = mod;
  }
  show() {
    return this.present() + ', it is a ' + this.model
  }
}
mycar = new Model("Ford", "Mustang");
mycar.show();
```

Arrow Function

```
hello = function() {
  return "Hello World!";
}
```

```
hello = () => {
  return "Hello World!";
}
```

React Rendor HTML

- React's goal is in many ways to render HTML in a web page.
- React renders HTML to the web page by using a function called ReactDOM.render().
- Unlike browser DOM elements, React elements are plain objects, and are cheap to create. React DOM takes care of updating the DOM to match the React elements.

```
ReactDOM.render(<p>Hello</p>, document.getElementById('root’));
<body>
  <div id="root"></div>
</body>
```

Root Node

- The root node is the HTML element where you want to display the result.
- It is like a container for content managed by React.
- It does NOT have to be a <div> element and it does NOT have to have the id='root’

<div id="root"></div>

- We call this a “root” DOM node because everything inside it will be managed by React DOM.
- Applications built with just React usually have a single root DOM node. If you are integrating React into an existing app, you may have as many isolated root DOM nodes as you like.
- To render a React element into a root DOM node, pass both to ReactDOM.render()

JSX

- JSX stands for JavaScript XML.
- JSX allows us to write HTML in React.
- JSX makes it easier to write and add HTML in React.
- JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.
- JSX converts HTML tags into react elements.
- You are not required to use JSX, but JSX makes it easier to write React applications.

```
const myelement = <h1>I Love JSX!</h1>;
ReactDOM.render(myelement, document.getElementById('root'));
```

React Components

- Components are like functions that return HTML elements.
- Components are independent and reusable bits of code.
- They serve the same purpose as JavaScript functions, but work in isolation and return HTML via a render() function.
- Conceptually, components are like JavaScript functions. They accept arbitrary inputs (called “props”) and return React elements describing what should appear on the screen.

Components come in two types:

- Class components and
- Function components

Class Component

- When creating a React component, the component's name must start with an upper case letter.
- The component has to include the extends React.Component statement, this statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.
- The component also requires a render() method, this method returns HTML.

```
class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}
```

Function Component

- A Function component also returns HTML, and behaves pretty much the same way as a Class component, but Class components have some additions.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

- This function is a valid React component because it accepts a single “props” (which stands for properties) object argument with data and returns a React element.

Props

- For handling component properties props can be used.
- Props are like function arguments, and you send them into the component as attributes.
- Props are arguments passed into React components.
- Props are passed to components via HTML attributes.
- React Props are like function arguments in JavaScript and attributes in HTML.

State

- React components has a built-in state object.
- The state object is where you store property values that belongs to the component.
- When the state object changes, the component re-renders.
- The state object is initialized in the constructor.

State

```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }
  render() {
    return (
      <div>
        <h1>My Car</h1>
      </div>
    );
  }
}
```

LifeCycle Methods

- Each component in React has a lifecycle which you can monitor and manipulate during its three main phases.

The three phases are:

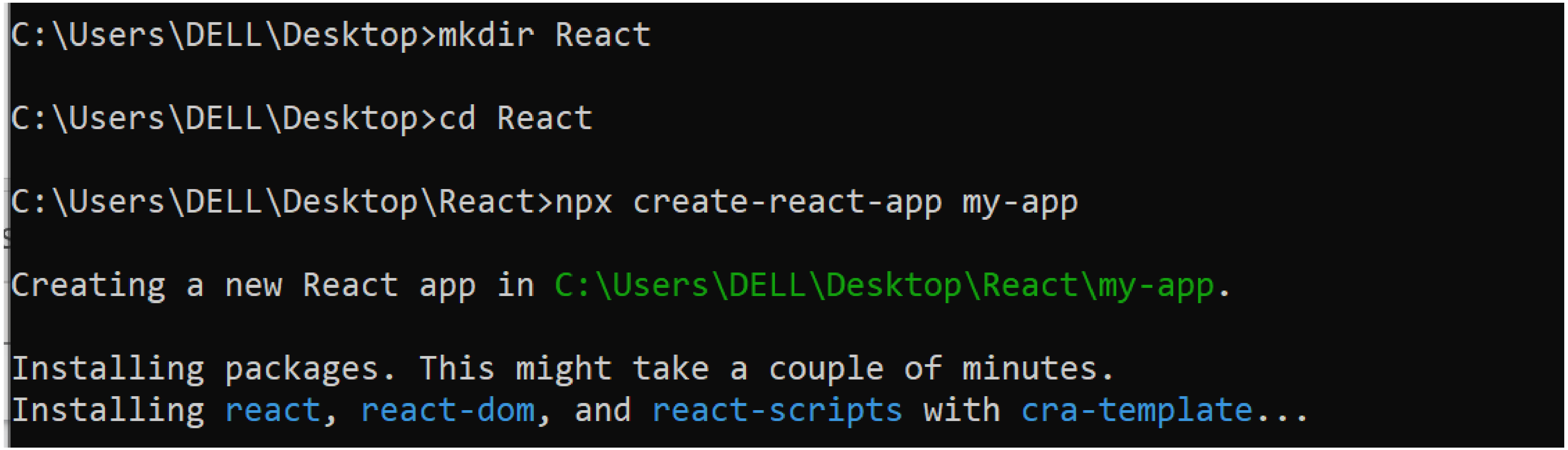
- Mounting,
- Updating, and
- Unmounting.

React JS

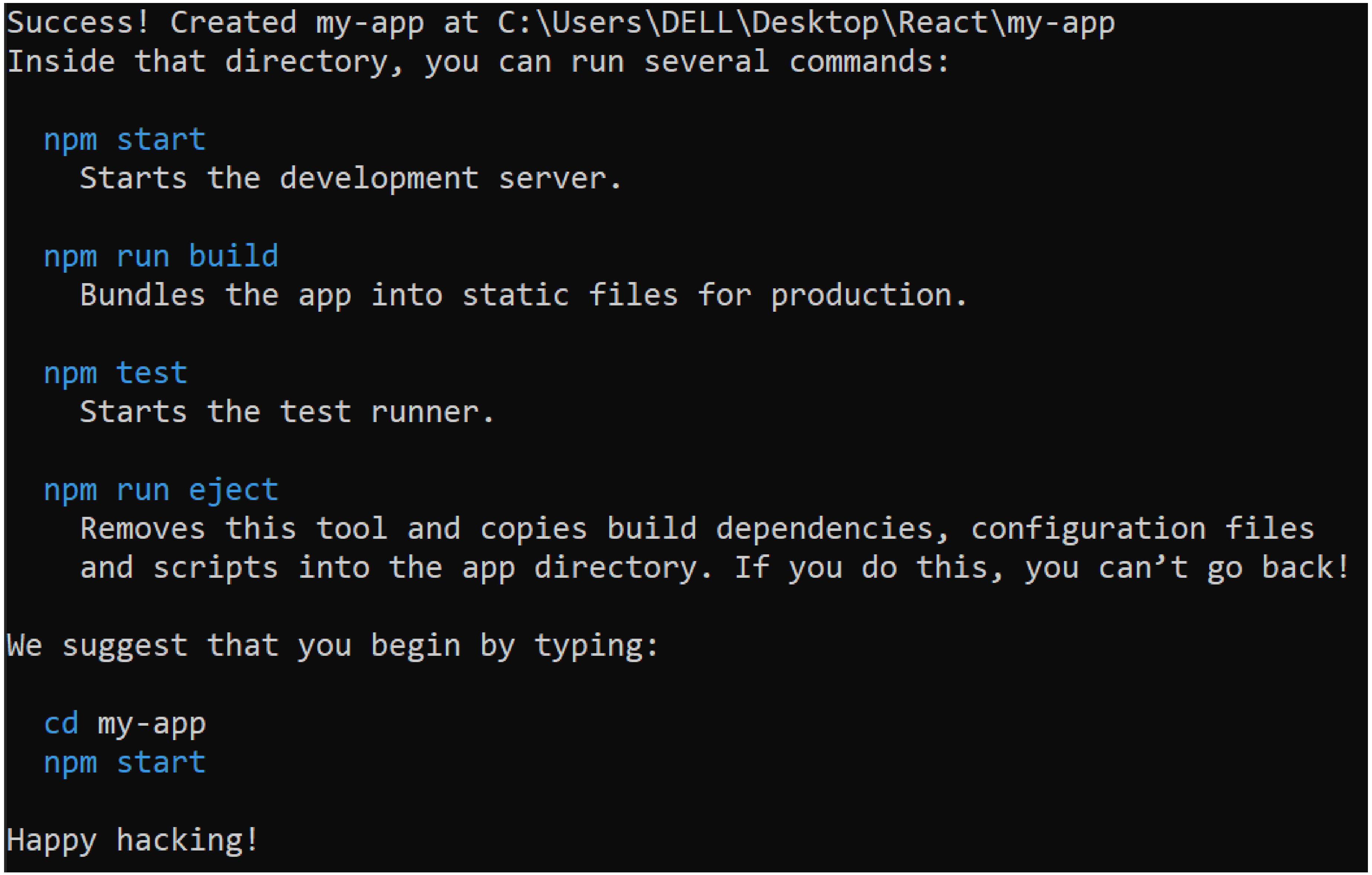
React JS

Installation

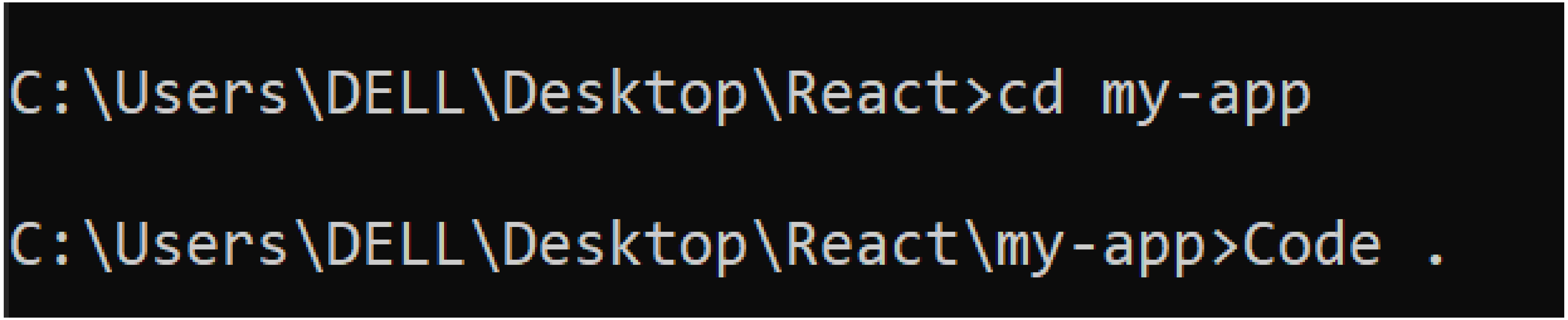
- Install Node js using the following link https://nodejs.org
- Npm(Node Package Manager) required for React is installed with Node.js.
- Check npm version using command “npm -v” on cmd window
- Create a Separate folder on desktop
- Move to that folder using cd command
- Then install React packages using the following command “npx create-react-app <app-name>”



```
C:\Users\DELL\Desktop>mkdir React

C:\Users\DELL\Desktop>cd React

C:\Users\DELL\Desktop\React>npx create-react-app my-app

Creating a new React app in C:\Users\DELL\Desktop\React\my-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
```



```
Success! Created my-app at C:\Users\DELL\Desktop\React\my-app
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

  cd my-app
  npm start

Happy hacking!
```



```
C:\Users\DELL\Desktop\React>cd my-app

C:\Users\DELL\Desktop\React\my-app>Code .
```

References

- https://www.w3schools.com
- https://nodejs.org
- https://www.youtube.com/watch?v=w7ejDZ8SWv8&t=10s&ab_channel=TraversyMedia
- https://reactjs.org/tutorial/tutorial.html
- https://www.w3schools.com/react/react_intro.asp
- https://reactjs.org/docs/hello-world.html