Introduction To Full-Stack Web Development

CS 386

Michael Kremer

Last updated: 11/30/2024 9:48:02 AM

- 28.1 Introduction to ReactJS
- 28.2 ReactJS Architecture
- 28.3 Getting started with ReactJS
- 28.4 ReactJS Concepts

Class 28

28.1 Introduction to ReactJS

- React was developed by Meta (formerly Facebook)
- ReactJS is open-source JavaScript library that is used for:
  - Building user interfaces in declarative and efficient way
- Component-based front-end library responsible only for:
  - View layer of MVC (Model View Controller) architecture
- React is used to create modular user interfaces
- Promotes development of reusable UI components that display dynamic data

28.1 Introduction to ReactJS

- ReactJS uses declarative paradigm:
  - ❑ Declarative view means:
    - Developers describe how the user interface should look based on its current state
    - Then React takes care of updating the DOM (Document Object Model) accordingly
  - ❑ Makes it possible for applications to be both effective and flexible
  - ❑ Creates simple views for each state in application
  - ❑ Efficiently updates and renders just right components when data changes
- Makes your code more predictable and easier to debug
- Each component in React application is responsible for:
  - ❑ Rendering separate, reusable piece of HTML code
- Ability to nest components within other components:
  - ❑ Allows for building of complex applications from simple building blocks
- Component may also keep track of its internal state:
  - ❑ Example: TabList component may keep variable for currently open tab in memory

28.1 Introduction to ReactJS

- Use Case:
  - Let us say one of your friends posted photograph on Facebook
  - Now you go and like image and then you started checking out comments too
  - Now while you are browsing over comments you see that likes count has increased by 100 since you liked picture, even without reloading the page
  - This magical count change is because of ReactJS

28.1 Introduction to ReactJS

How does it work?

- While building client-side apps, team of Facebook developers realized that DOM is slow
- Document Object Model (DOM) is application programming interface (API) for HTML and XML documents
- Defines logical structure of documents and way document is accessed and manipulated
- To make it faster, React implements virtual DOM:
  - DOM tree representation in JavaScript
  - When it needs to read or write to DOM, it will use the virtual representation of it
  - Virtual DOM will try to find most efficient way to update browser's DOM

28.1 Introduction to ReactJS

How does it work?

- 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:
  - Reason for this is that JavaScript is very fast
  - Worth keeping DOM tree in it to speed up its manipulation
- React was developed to be used in browser
  - But because of its design it can also be used in server with Node.js

28.2 ReactJS Architecture

JSX (JavaScript Syntax Extension)

- JavaScript XML is abbreviated as JSX (Syntax extension for JavaScript)
- ReactJS uses XML or HTML-like syntax
- Syntax is turned into React Framework JavaScript calls
- Expands ES6 to allow HTML-like text to coexist with JavaScript react code
- Not required to use JSX, but it is recommended in ReactJS
- JSX is JavaScript code with combination of XML syntax in it
- JSX tag has tag name, attributes and children which make it look like XML

28.2 ReactJS Architecture

JSX and Babel

- ➢ JSX cannot be implemented directly by browsers
- ➢ Instead requires "compiler" to transform it into ECMAScript → Babel comes in
- ➢ Babel acts as this "compiler" (transpiler) allowing to leverage all benefits of JSX while building React components
- ➢ Babel's use is not only rooted in React:
  - ❑ Main purpose is as compiler to convert code written in ECMAScript2015+ into backwards-compatible JavaScript
  - ❑ As JavaScript continues to advance, Babel makes available syntax transformer plugins so users may employ latest syntax in their code without having to wait for browser support

28.2 ReactJS Architecture

Virtual DOM

- ➢ Most crucial aspect of web development since it splits the code into modules and then executes it
- ➢ JavaScript frameworks typically update entire DOM at once, making the online app slow
- ➢ React makes use of virtual DOM, which is carbon copy of real DOM
- ➢ When web application is modified:
  - ❑ Virtual DOM is updated first
  - ❑ Difference between real DOM and virtual DOM is determined
- ➢ When it discovers the difference:
  - ❑ DOM updates parts that have changed recently
  - ❑ Leaving rest unchanged

28.2 ReactJS Architecture

One-way Data Binding

- ➢ As name implies, one-way data binding is one-direction or unidirectional flow
- ➢ Data only goes in one direction:
  - ❑ From top to bottom, from parent components to child components
  - ❑ Child component's properties (props) cannot return data to its parent component
  - ❑ But they can communicate with it to change states based on inputs
- ➢ One-way data-binding operates in this manner
- ➢ Everything remains modular, and as result, quick

28.2 ReactJS Architecture

Declarative

- ReactJS uses simple JavaScript:
  - ❑ To enable component-based approach to develop websites and mobile apps
- Benefit of cutting development costs
- Best features of ReactJS enable web pages and mobile apps to have highly interactive and dynamic user interfaces
- When your data changes:
  - ❑ Creates basic views for each project state
  - ❑ React will update and render only relevant components
- When used frequently on websites and apps, library becomes easier

28.2 ReactJS Architecture

React Native

- ReactJS is used to build user interfaces of web applications (apps running in web browser)
- React Native is used to build applications that run on both iOS and Android devices (that is, cross-platform mobile applications)
- Instead of using web components like ReactJS, React Native employs native components as building blocks
- To get started with React Native, need to understand basic React concepts:
  - JSX
  - Components
  - State
  - Props
- Even if already familiar with React, need to learn about React Native capabilities like native components

28.2 ReactJS Architecture

Component-Based

- Everything in React is web page component separated into individual components to form view (or UI)
- Each visual part of software is encapsulated within component, which is self-contained module
- Because component functionality is defined in JavaScript rather than templates:
    - Easily pass rich data through your app and keep state out of the DOM
- ReactJS components:
    - Building blocks of any React application
    - One of best features in ReactJS
- Single app is typically built of numerous components
- User interface element is essentially most significant component
- React separates user interface into reusable components that may be processed separately

28.3 Getting started with ReactJS

- Basic example to get started with ReactJS
- Uses Babel script inside HTML page
- Once Babel script is stored in separate file → does not work anymore with file protocol
- CORS error → Prevents loading files from filesystem
- Need to use local webserver to use http protocol
- To get React CDNs:
  - https://legacy.reactjs.org/docs/cdn-links.html
- Script tag used in HTML page using type text/babel

28.3 Getting started with ReactJS

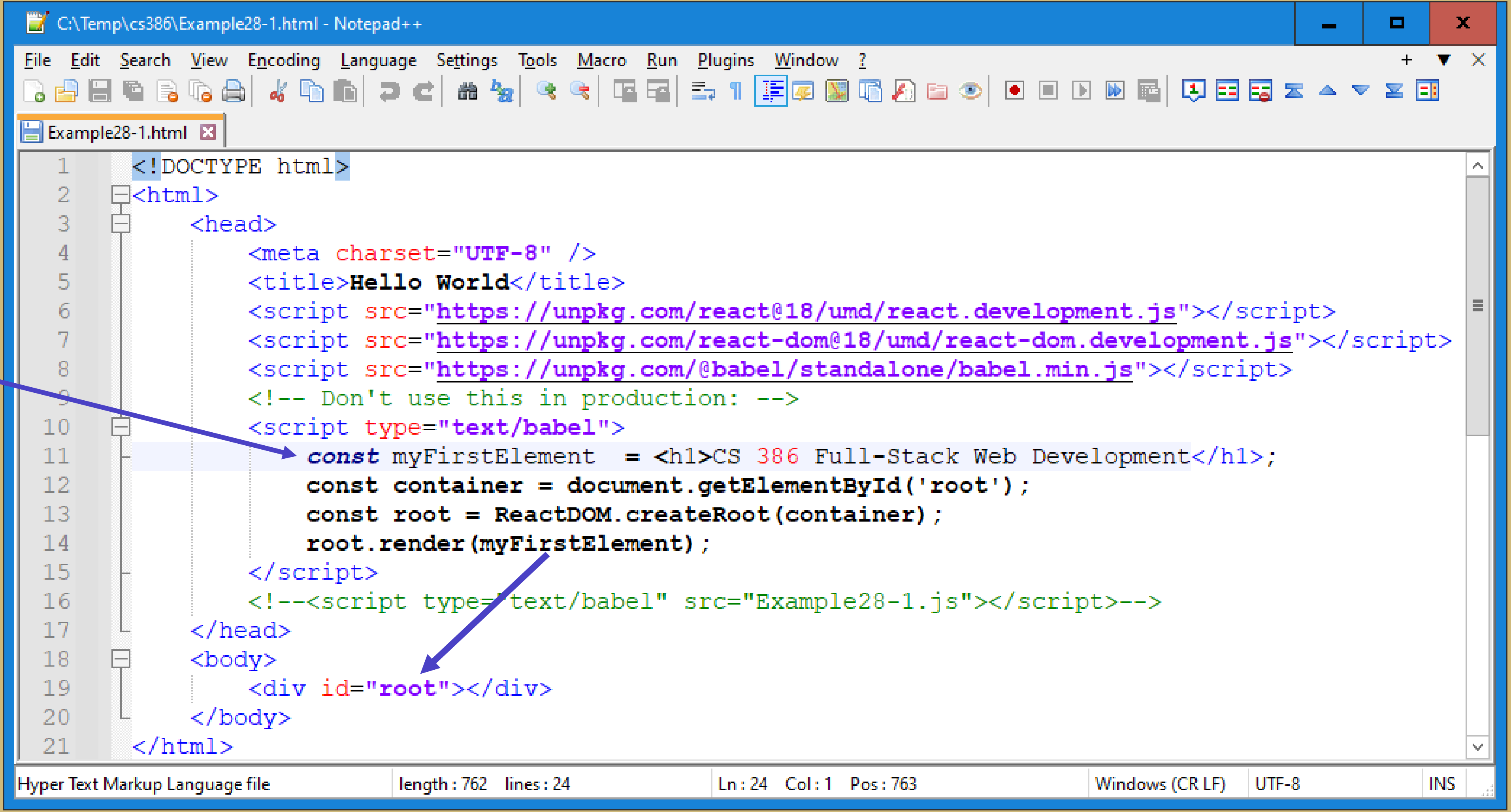
Example 28-1:

- Use file Example28-1.html
- Create script tag in html head section with type “text/babel”
- Create constant myFirstElement
  - Assign h1 header with content (no quotes!!): CS386 Full-Stack Web Development
- Create constant container
  - Assign get element by id ‘root’
- Create constant root
  - Use createRoot method of ReactDOM passing container
- Use method render on root passing function myFirstElement as html markup
- Notice html markup is not string!

28.3 Getting started with ReactJS

Example 28-1:

- Write following code in between script tags:
- Notice html markup is not string!



C:\Temp\cs386\Example28-1.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Example28-1.html

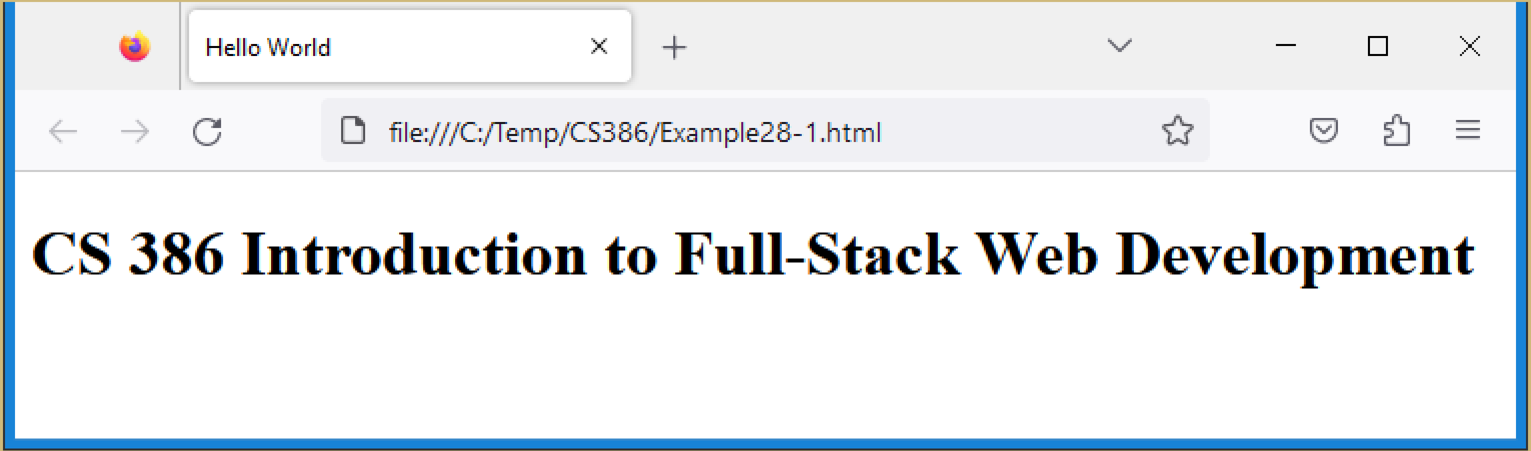
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>Hello World</title>
        <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
        <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
        <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
        <!-- Don't use this in production: -->
        <script type="text/babel">
            const myFirstElement  = <h1>CS 386 Full-Stack Web Development</h1>;
            const container = document.getElementById('root');
            const root = ReactDOM.createRoot(container);
            root.render(myFirstElement);
        </script>
        <!--<script type="text/babel" src="Example28-1.js"></script>-->
    </head>
    <body>
        <div id="root"></div>
    </body>
</html>
```

Hyper Text Markup Language file
length : 762 lines : 24
Ln : 24 Col : 1 Pos : 763
Windows (CR LF)
UTF-8
INS

28.3 Getting started with ReactJS

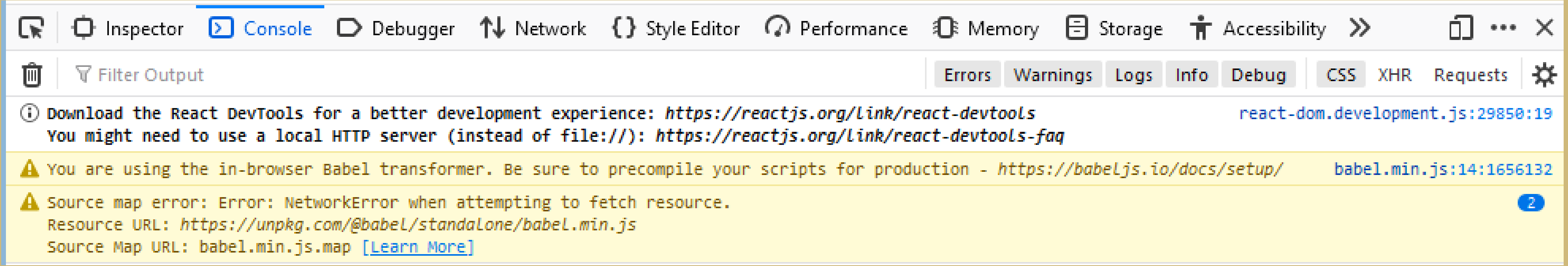
Example 28-1:

- You noticed that it takes some time to load html page
- JSX compilation takes place



CS 386 Introduction to Full-Stack Web Development

- Open web developer tools:

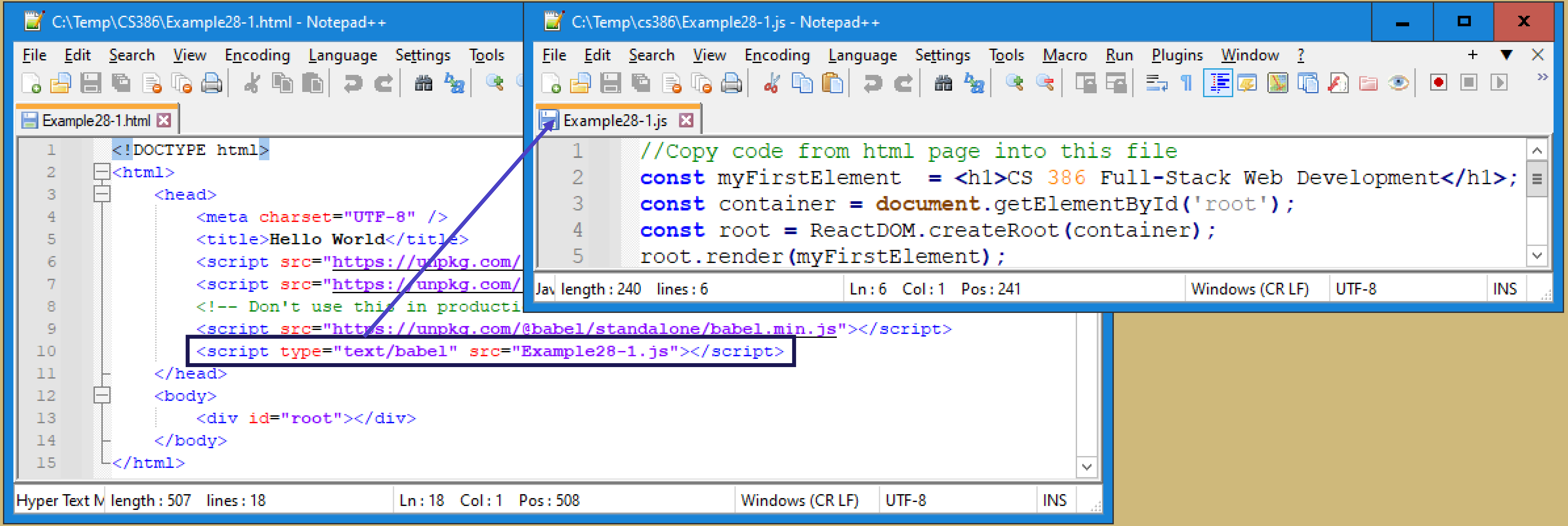


```
ⓘ Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools    react-dom.development.js:29850:19
  You might need to use a local HTTP server (instead of file://): https://reactjs.org/link/react-devtools-faq
⚠ You are using the in-browser Babel transformer. Be sure to precompile your scripts for production - https://babeljs.io/docs/setup/    babel.min.js:14:1656132
⚠ Source map error: Error: NetworkError when attempting to fetch resource.    2
  Resource URL: https://unpkg.com/@babel/standalone/babel.min.js
  Source Map URL: babel.min.js.map [Learn More]
```

28.3 Getting started with ReactJS

Example 28-1:

- Place Babel script in separate JavaScript file (Example28-1.js) (better way to do it)
- Remove script from html page, instead load Example28-1.js



C:\Temp\CS386\Example28-1.html - Notepad++

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>Hello World</title>
        <script src="https://unpkg.com/
        <script src="https://unpkg.com/
        <!-- Don't use this in producti
        <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
        <script type="text/babel" src="Example28-1.js"></script>
    </head>
    <body>
        <div id="root"></div>
    </body>
</html>
```

Hyper Text M length : 507 lines : 18 Ln : 18 Col : 1 Pos : 508 Windows (CR LF) UTF-8 INS

C:\Temp\cs386\Example28-1.js - Notepad++

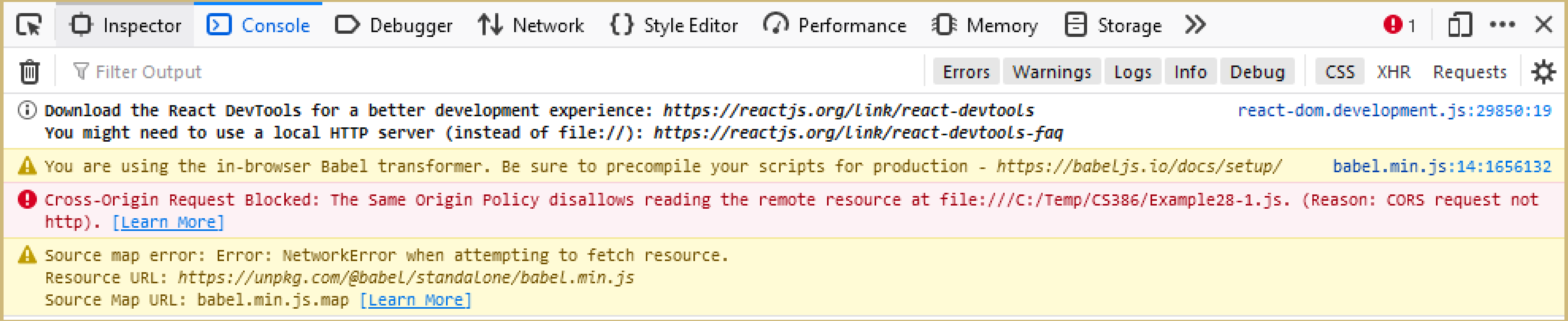
```javascript
//Copy code from html page into this file
const myFirstElement  = <h1>CS 386 Full-Stack Web Development</h1>;
const container = document.getElementById('root');
const root = ReactDOM.createRoot(container);
root.render(myFirstElement);
```

Jav length : 240 lines : 6 Ln : 6 Col : 1 Pos : 241 Windows (CR LF) UTF-8 INS

28.3 Getting started with ReactJS

Example 28-1:

- HTML page is not rendered anymore
- Notice error in console

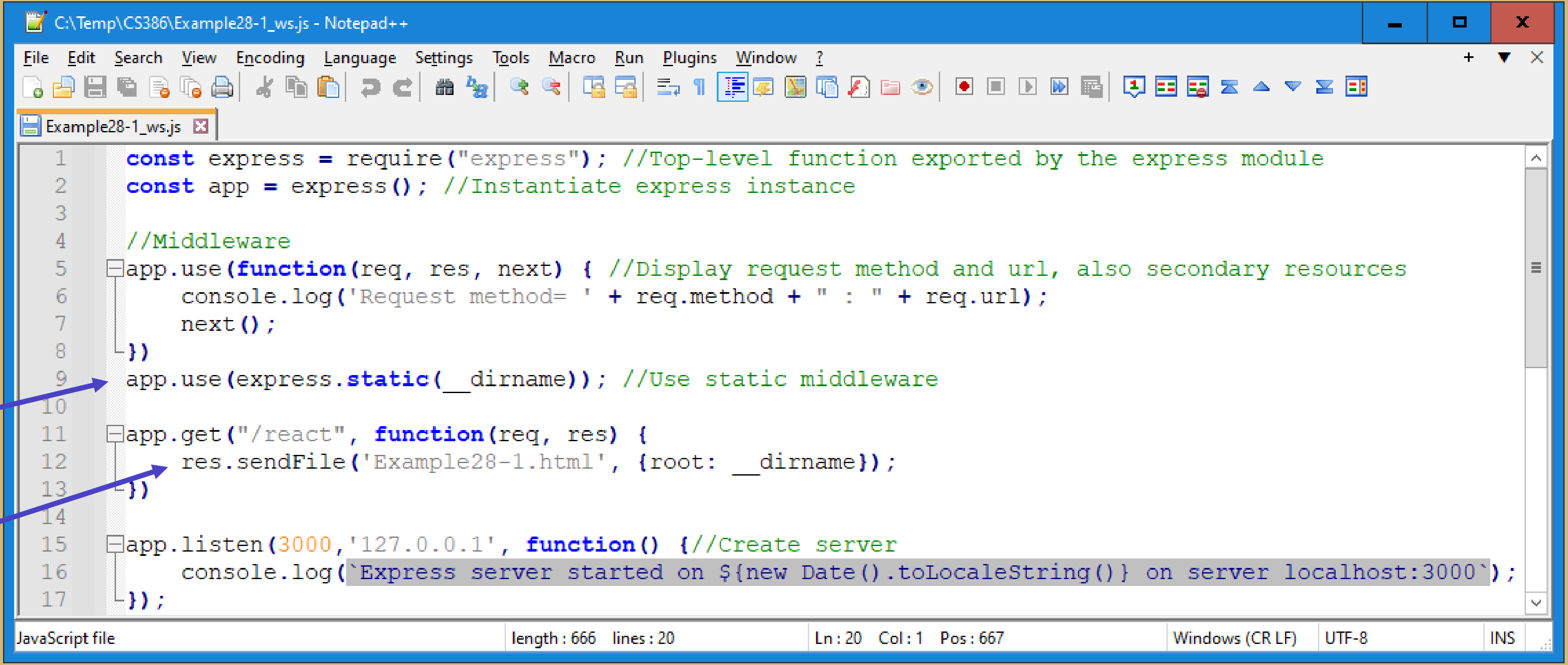


```
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools     react-dom.development.js:29850:19
You might need to use a local HTTP server (instead of file://): https://reactjs.org/link/react-devtools-faq
You are using the in-browser Babel transformer. Be sure to precompile your scripts for production - https://babeljs.io/docs/setup/     babel.min.js:14:1656132
Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at file:///C:/Temp/CS386/Example28-1.js. (Reason: CORS request not
http). [Learn More]
Source map error: Error: NetworkError when attempting to fetch resource.
Resource URL: https://unpkg.com/@babel/standalone/babel.min.js
Source Map URL: babel.min.js.map [Learn More]
```

28.3 Getting started with ReactJS

Example 28-1:

- Need to use web server in order to use http
- Use simple node web server:
  - Example28-1_ws.js
- Uses static middleware for current folder
- Route react will serve file

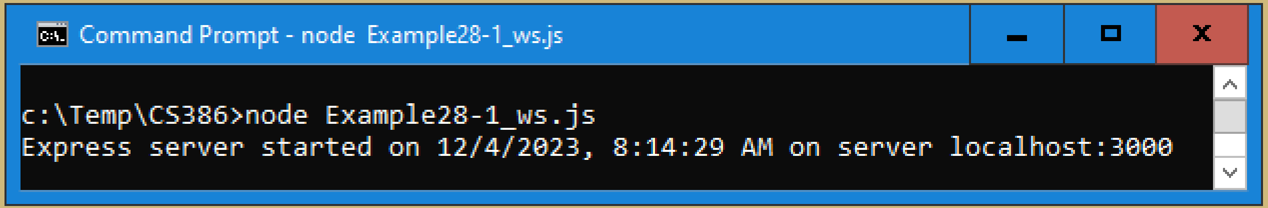

C:\Temp\CS386\Example28-1_ws.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Example28-1_ws.js

```
const express = require("express"); //Top-level function exported by the express module
const app = express(); //Instantiate express instance

//Middleware
app.use(function(req, res, next) { //Display request method and url, also secondary resources
    console.log('Request method= ' + req.method + " : " + req.url);
    next();
})
app.use(express.static(__dirname)); //Use static middleware

app.get("/react", function(req, res) {
    res.sendFile('Example28-1.html', {root: __dirname});
})

app.listen(3000,'127.0.0.1', function() {//Create server
    console.log(`Express server started on ${new Date().toLocaleString()} on server localhost:3000`);
});
```

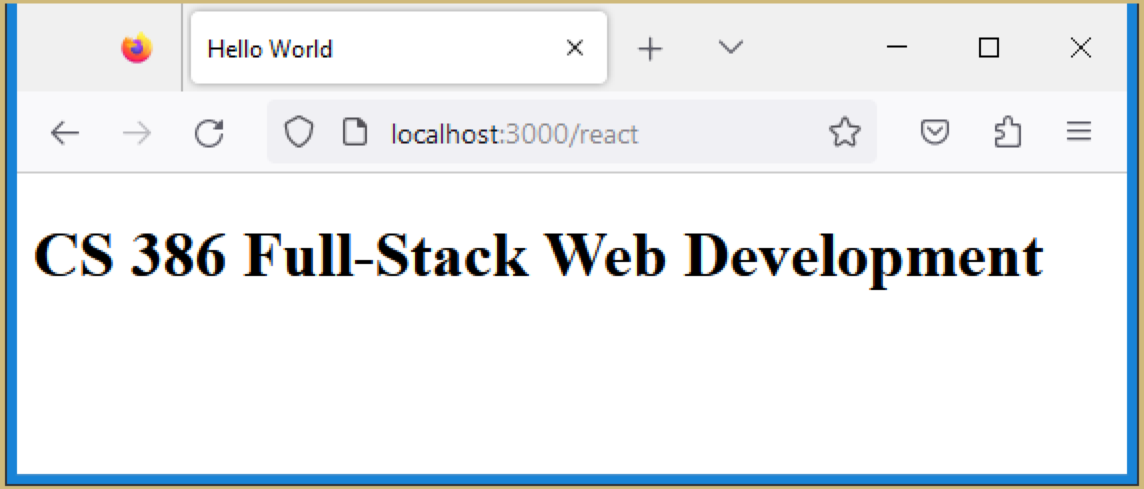
28.3 Getting started with ReactJS

Example 28-1:

- ➢ Execute web server in node:
  - ❑ node Example28-1_ws.js



```
c:\Temp\CS386>node Example28-1_ws.js
Express server started on 12/4/2023, 8:14:29 AM on server localhost:3000
```



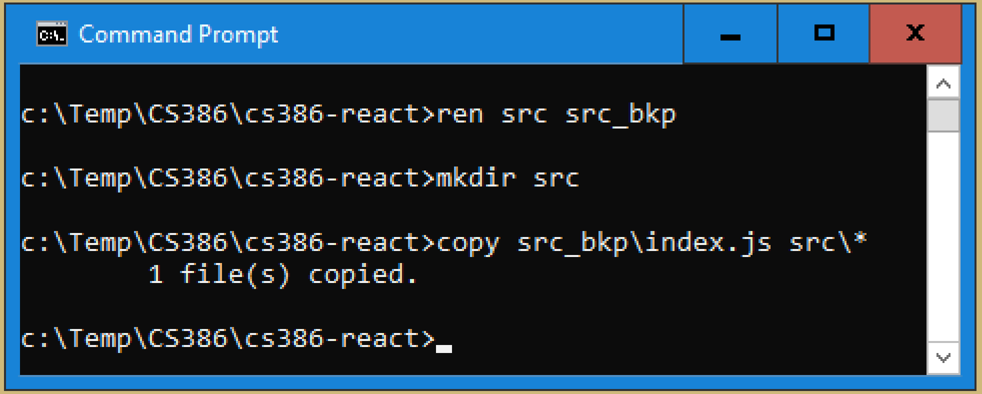
CS 386 Full-Stack Web Development

28.3 Getting started with ReactJS

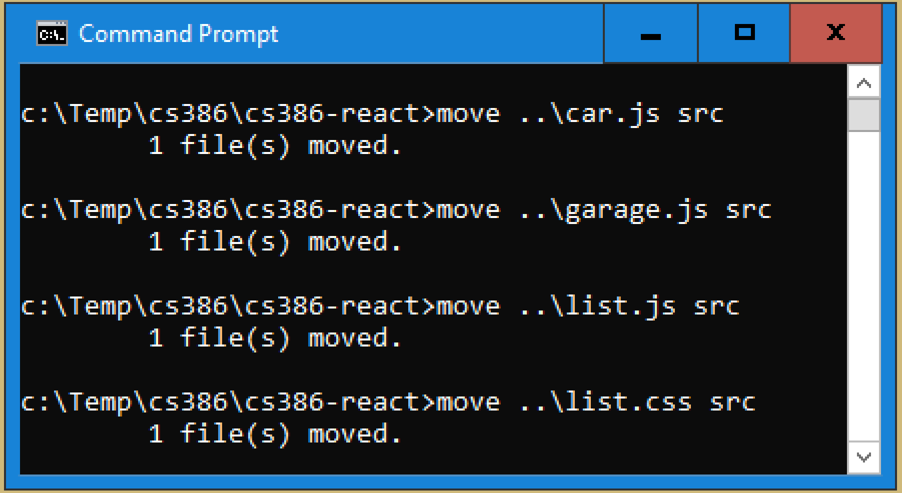
- Set up react development environment:
  - In console/terminal, navigate to CS386 folder
  - Execute following command: npx create-react-app cs386-react
- What is NPX?
  - NPX stands for Node Package eXecute
  - Simple NPM package runner
  - Allows developers to execute any Javascript Package available on NPM registry without even installing it
- To start the development environment:
  - Navigate to folder CS386/cs386-react
  - Execute following command: npm start (this may take some time)

28.3 Getting started with ReactJS

- Rename folder src → src_bkp (to keep original files)
- Create new folder src
- Copy file index.js from src_bkp into src
- Windows:
  - ren src src_bkp
  - mkdir src
  - copy src_bkp\index.js src*
  - Also move following files into src folder:
    - move ..\car.js src
    - move ..\garage.js src
    - move ..\list.js src
    - move ..\list.css src
- Linux/Mac on next slide



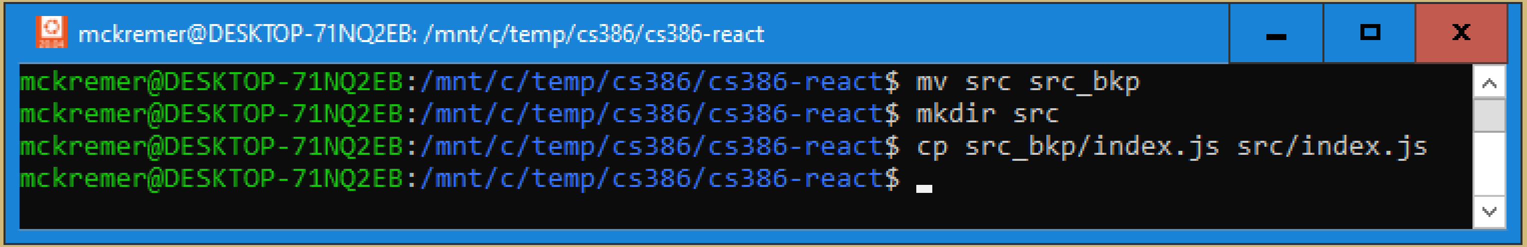
```
c:\Temp\CS386\cs386-react>ren src src_bkp

c:\Temp\CS386\cs386-react>mkdir src

c:\Temp\CS386\cs386-react>copy src_bkp\index.js src\*
        1 file(s) copied.

c:\Temp\CS386\cs386-react>_
```



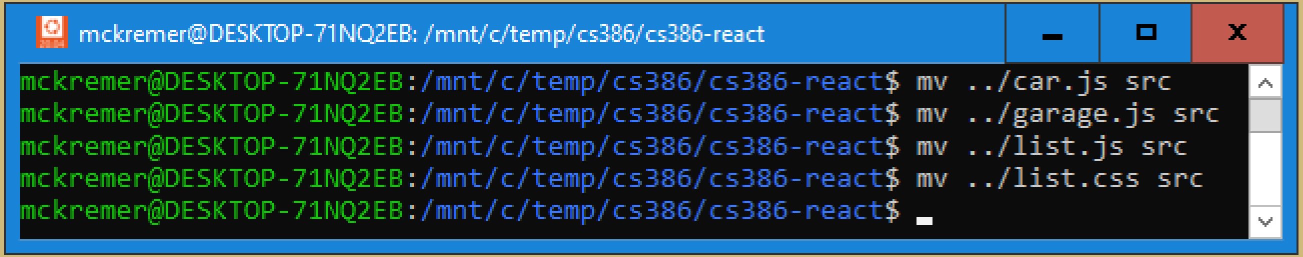
```
c:\Temp\cs386\cs386-react>move ..\car.js src
        1 file(s) moved.

c:\Temp\cs386\cs386-react>move ..\garage.js src
        1 file(s) moved.

c:\Temp\cs386\cs386-react>move ..\list.js src
        1 file(s) moved.

c:\Temp\cs386\cs386-react>move ..\list.css src
        1 file(s) moved.
```

28.3 Getting started with ReactJS

- ➢ Rename folder src → src_bkp (to keep original files)
- ➢ Create new folder src
- ➢ Copy file index.js from src_bkp into src
- ➢ Linux/Mac:
  - ❑ mv src src_bkp
  - ❑ mkdir src
  - ❑ cp src_bkp/index.js src/index.js
  - ❑ Also move following files into src folder:
    - o mv ../car.js src
    - o mv ../garage.js src
    - o mv ../list.js
    - o mv ../list.css



```
mckremer@DESKTOP-71NQ2EB: /mnt/c/temp/cs386/cs386-react
mckremer@DESKTOP-71NQ2EB:/mnt/c/temp/cs386/cs386-react$ mv src src_bkp
mckremer@DESKTOP-71NQ2EB:/mnt/c/temp/cs386/cs386-react$ mkdir src
mckremer@DESKTOP-71NQ2EB:/mnt/c/temp/cs386/cs386-react$ cp src_bkp/index.js src/index.js
mckremer@DESKTOP-71NQ2EB:/mnt/c/temp/cs386/cs386-react$ _
```

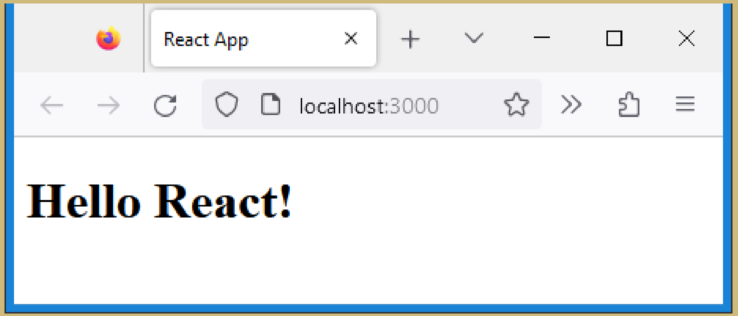


```
mckremer@DESKTOP-71NQ2EB: /mnt/c/temp/cs386/cs386-react
mckremer@DESKTOP-71NQ2EB:/mnt/c/temp/cs386/cs386-react$ mv ../car.js src
mckremer@DESKTOP-71NQ2EB:/mnt/c/temp/cs386/cs386-react$ mv ../garage.js src
mckremer@DESKTOP-71NQ2EB:/mnt/c/temp/cs386/cs386-react$ mv ../list.js src
mckremer@DESKTOP-71NQ2EB:/mnt/c/temp/cs386/cs386-react$ mv ../list.css src
mckremer@DESKTOP-71NQ2EB:/mnt/c/temp/cs386/cs386-react$ _
```

28.3 Getting started with ReactJS

Example 28-2:

- Use file Example 28-2.js
- Imports are already done
- Create h1 header with some tile, assign into constant myFirstElement
  - **IMPORTANT:** Do not enclose in quotes!!!!
- Use method createRoot on ReactDOM:
  - Passing reference of element with id 'root'
  - Assign into constant root
- Use method render on root passing myFirstElement

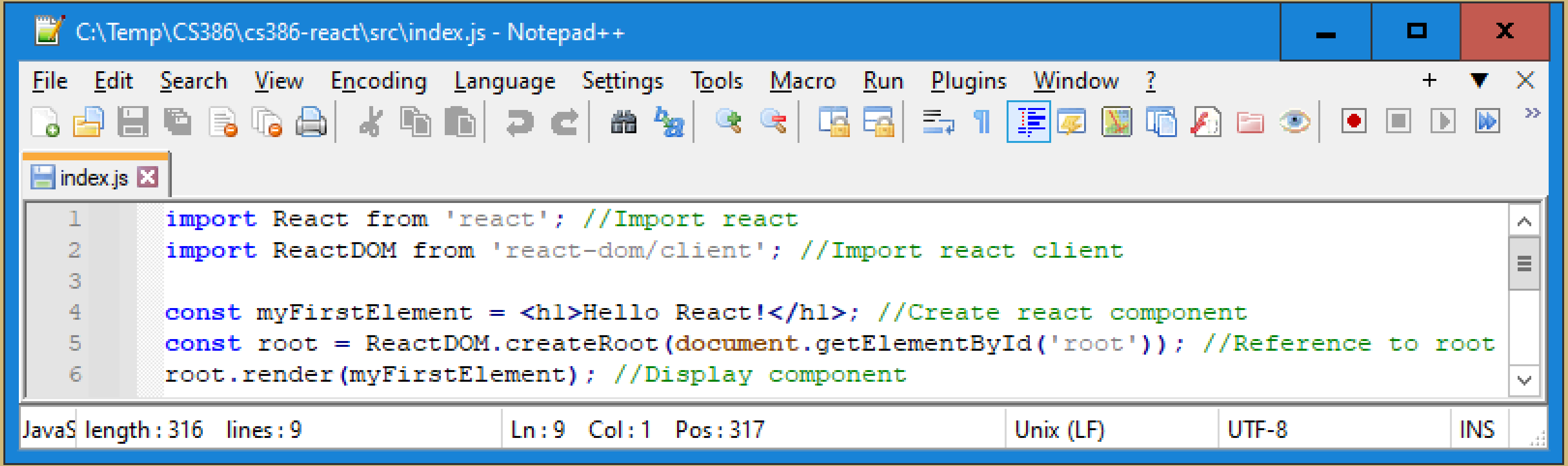


Hello React!

28.3 Getting started with ReactJS

Example 28-2:

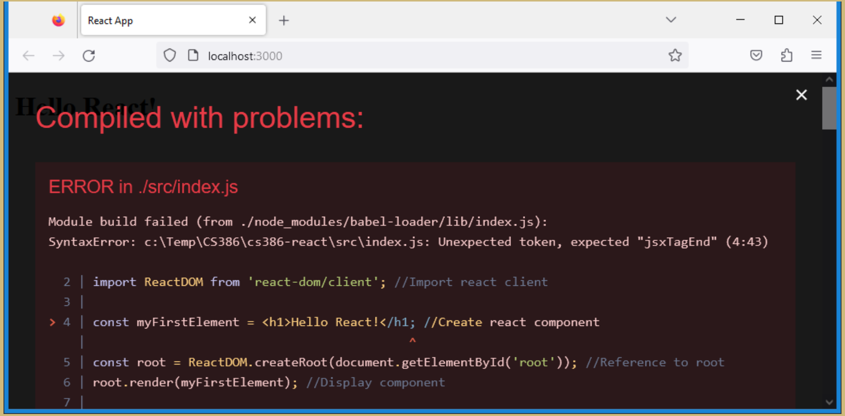
- Copy and paste code from Example28-2.js into index.js in src folder


C:\Temp\CS386\cs386-react\src\index.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
index.js

```
import React from 'react'; //Import react
import ReactDOM from 'react-dom/client'; //Import react client

const myFirstElement = <h1>Hello React!</h1>; //Create react component
const root = ReactDOM.createRoot(document.getElementById('root')); //Reference to root
root.render(myFirstElement); //Display component
```

JavaS length : 316 lines : 9 Ln : 9 Col : 1 Pos : 317 Unix (LF) UTF-8 INS

- Create an error, remove closing angle bracket from closing h1 tag:
- Notice error message in browser

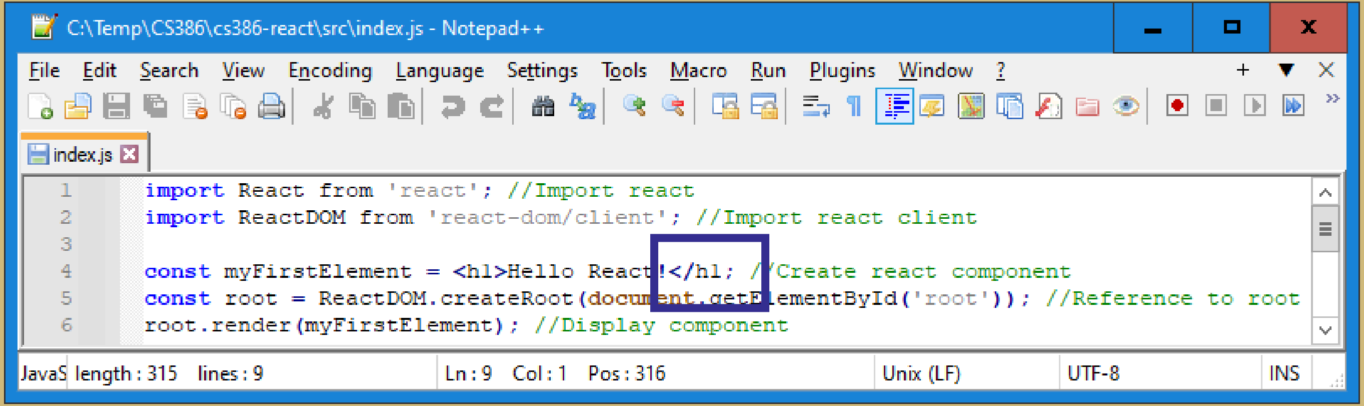


Hello React!

Compiled with problems:

ERROR in ./src/index.js

```
Module build failed (from ./node_modules/babel-loader/lib/index.js):
SyntaxError: c:\Temp\CS386\cs386-react\src\index.js: Unexpected token, expected "jsxTagEnd" (4:43)

  2 | import ReactDOM from 'react-dom/client'; //Import react client
  3 |
> 4 | const myFirstElement = <h1>Hello React!</h1; //Create react component
    |                                           ^
  5 | const root = ReactDOM.createRoot(document.getElementById('root')); //Reference to root
  6 | root.render(myFirstElement); //Display component
  7 |
```


C:\Temp\CS386\cs386-react\src\index.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
index.js

```
import React from 'react'; //Import react
import ReactDOM from 'react-dom/client'; //Import react client

const myFirstElement = <h1>Hello React!</h1; //Create react component
const root = ReactDOM.createRoot(document.getElementById('root')); //Reference to root
root.render(myFirstElement); //Display component
```

JavaS length : 315 lines : 9 Ln : 9 Col : 1 Pos : 316 Unix (LF) UTF-8 INS

28.4 ReactJS Concepts

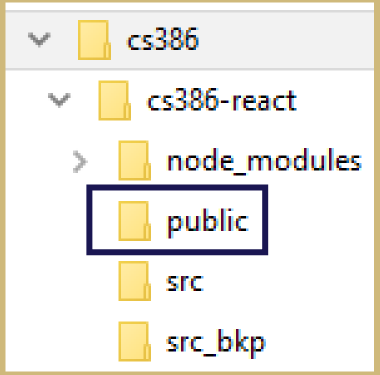
ReactJS uses JavaScript 2015 (ES6)

- The following are new features in JavaScript ES6:
  - Classes
  - Arrow Functions
  - **Variables (let, const, var)**
  - **Array Methods like .map()**
  - Destructuring
  - **Modules**
  - **Ternary Operator**
  - Spread Operator
- Already covered few new features in this course (in bold)
- Learn few more now with ReactJS

28.4 ReactJS Concepts

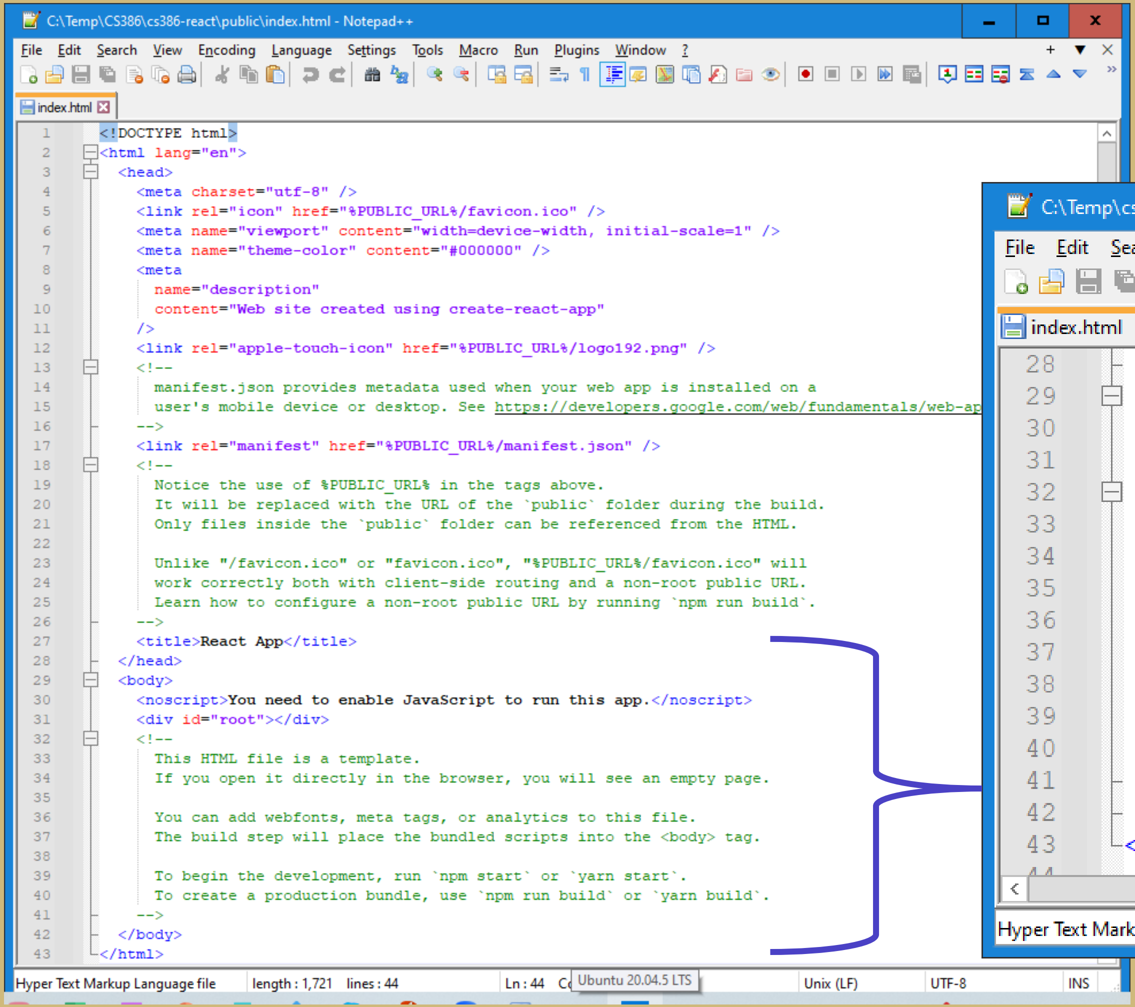
React Render HTML

- React's goal is in many ways to render HTML in web pages
- React renders HTML to web page by using:
  - Function called createRoot() and its method render()
- createRoot Function:
  - createRoot() function takes one argument: HTML element
  - Purpose of function is to define HTML element where React component should be displayed
- render Method:
  - render() method is then called to define React component that should be rendered
  - But render where?
- Another folder in root directory of React project, named "public“:
  - In this folder, there is index.html file
  - Notice single <div> in body of this file with id=“root”
  - This is where React application will be rendered



28.4 ReactJS Concepts

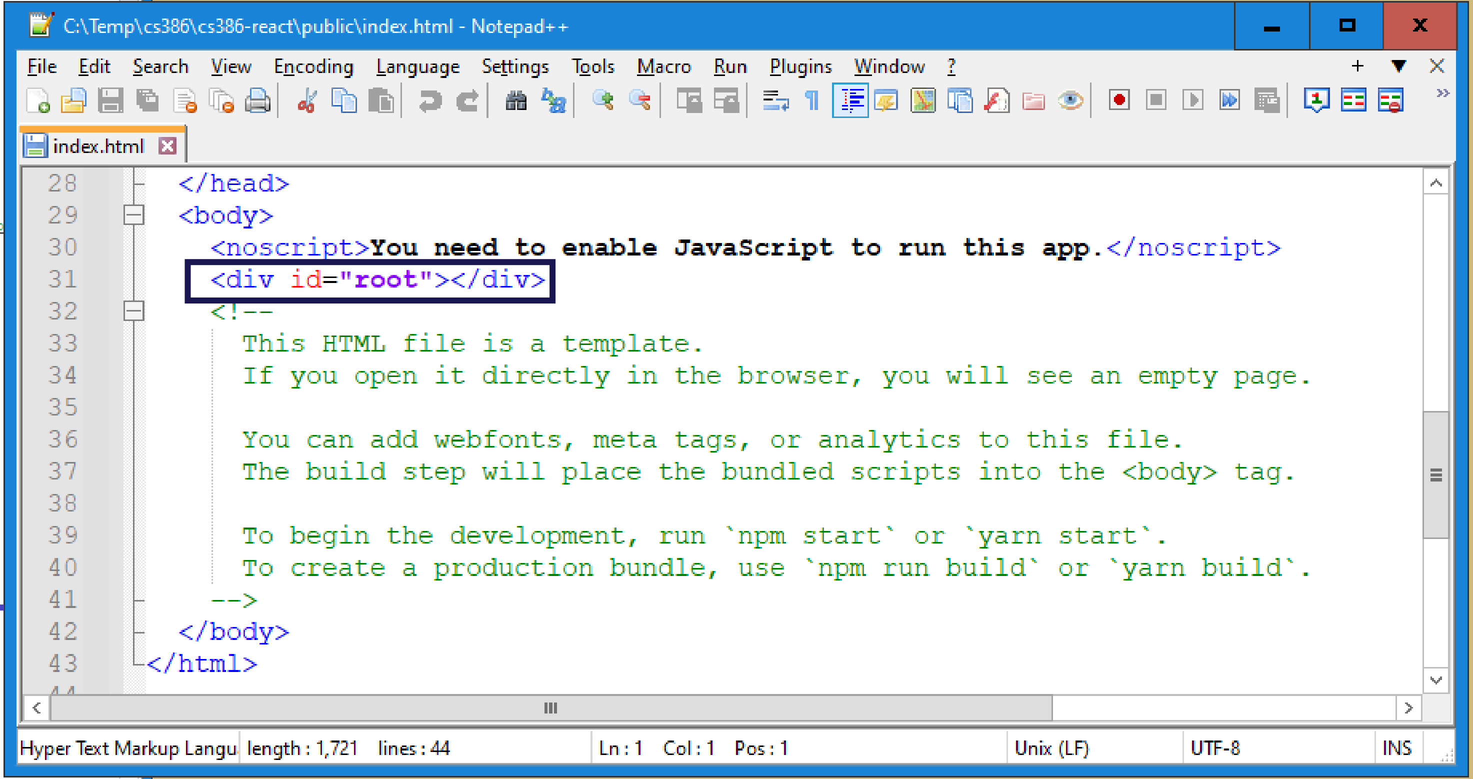
- ReactJS index.html



C:\Temp\CS386\cs386-react\public\index.html - Notepad++

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-ap
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

Hyper Text Markup Language file | length : 1,721 lines : 44 | Ln : 44 | Ubuntu 20.04.5 LTS | Unix (LF) | UTF-8 | INS



C:\Temp\cs386\cs386-react\public\index.html - Notepad++

```html
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

Hyper Text Markup Langu | length : 1,721 lines : 44 | Ln : 1 Col : 1 Pos : 1 | Unix (LF) | UTF-8 | INS

28.4 ReactJS Concepts

Root Node

- ➢ root node is HTML element where to display result
- ➢ Container for content managed by React:
    - ❑ Does NOT have to be <div> element
    - ❑ Does NOT have to have id='root'
    - ❑ But this is standard convention

28.4 ReactJS Concepts

HTML Code

- HTML code in previous examples used JSX
- Allows you to write HTML tags inside JavaScript code
- Note that html code is not string but assigned into variable as html code

28.4 ReactJS Concepts

JSX

- What is JSX?
  - JSX stands for JavaScript **X**ML/E**x**tension
  - JSX allows to write HTML in React
  - JSX makes it easier to write and add HTML in React
- Coding JSX
  - JSX allows:
    - To write HTML elements in JavaScript
    - Place them in DOM without any createElement() and/or appendChild() methods
  - JSX converts HTML tags into react elements:
    - Not required to use JSX
    - But JSX makes it easier to write React applications

28.4 ReactJS Concepts

Syntax:
React.**createElement(***type*, *props*, **...***children***)**

JSX

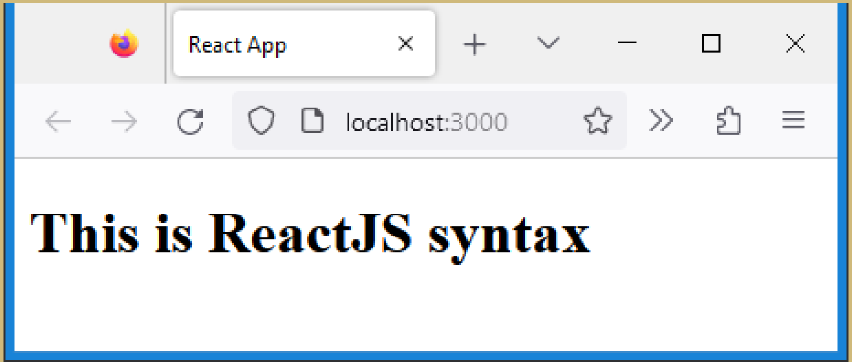
- React **without** JSX:
  - ❑ Use createElement in ReactJS to create html element **without** JSX
- type:
  - ❑ Either valid React component type (tag name string (such as 'div' or 'span’)
  - ❑ Or React component, such as function, class, or special component like Fragment
- props:
  - ❑ Must either be object or null, if null, it will be treated same as empty object
  - ❑ React will create element with props matching props you have passed
  - ❑ Represent properties like id, class, onClick, etc
- optional ...children:
  - ❑ Zero or more child nodes
  - ❑ Can be any React nodes, including React elements, strings, numbers, portals, empty nodes (null, undefined, true, and false), and arrays of React nodes

28.4 ReactJS Concepts

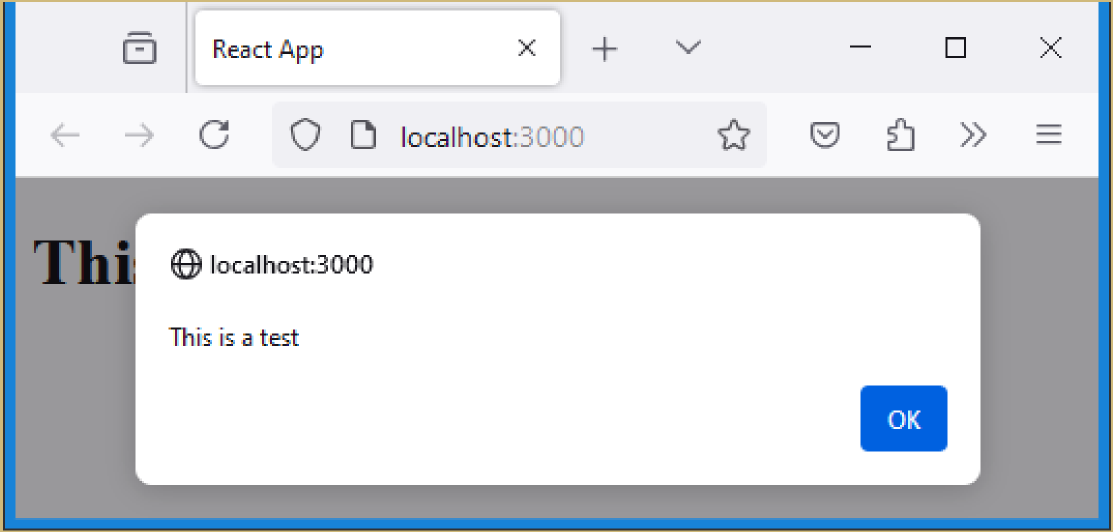
Example 28-3:

Syntax:
import *React* **from 'react';**
*React***.createElement(***type*, *props*, **...***children***)**

- Only replace line with JSX syntax in previous example, everything else remains the same
- Using React method createElement without JSX, assign into const myElement:
  - type: 'h1'
  - props: {onClick: function(){alert('This is a test');}}
  - children: 'This is ReactJS syntax



This is ReactJS syntax



This is a test

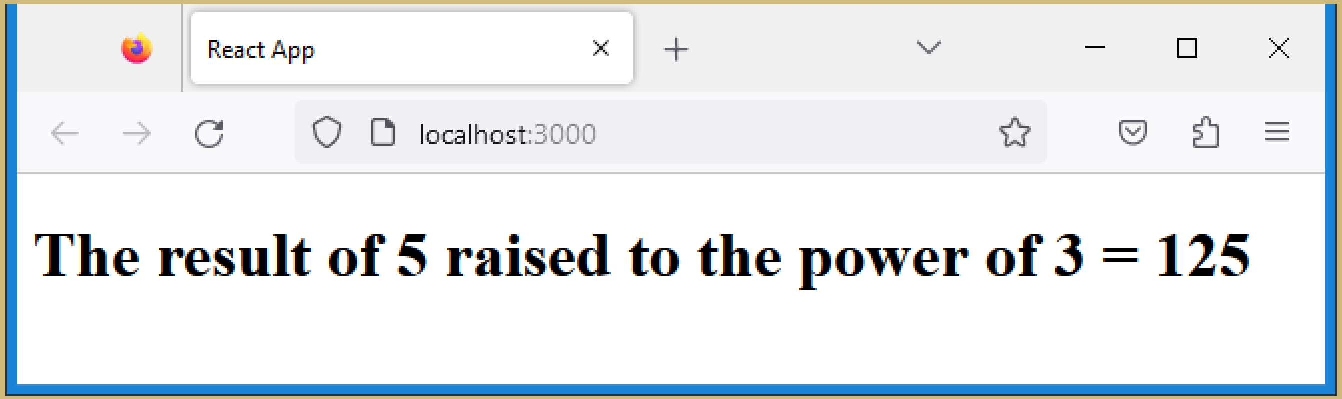
28.4 ReactJS Concepts

Expressions in JSX

- ➢ With JSX you can write expressions inside curly braces { }
- ➢ Expression can be:
  - ❑ React variable
  - ❑ Property
  - ❑ Any other valid JavaScript expression
- ➢ JSX will execute expression and return the result

Example 28-4:

- ➢ Using expressions enclosed in curly braces

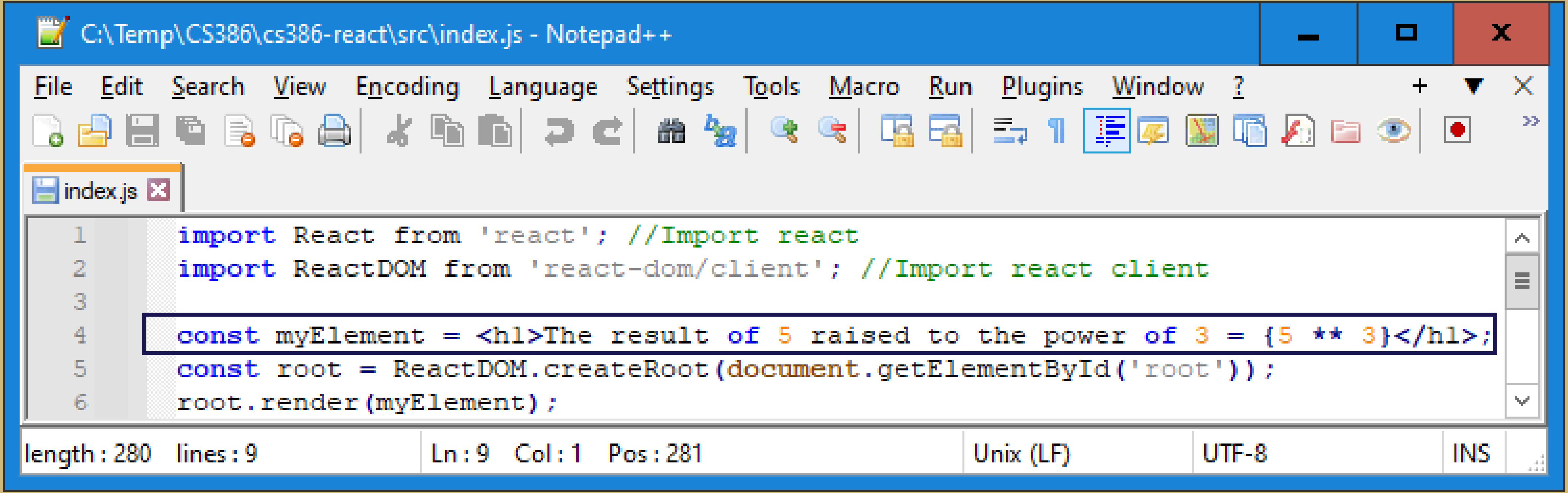


```
The result of 5 raised to the power of 3 = 125
```

28.4 ReactJS Concepts

Example 28-4:

➢ Replace only JSX line creating h1 header with expression


C:\Temp\CS386\cs386-react\src\index.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
index.js

```
import React from 'react'; //Import react
import ReactDOM from 'react-dom/client'; //Import react client

const myElement = <h1>The result of 5 raised to the power of 3 = {5 ** 3}</h1>;
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

length : 280   lines : 9   Ln : 9   Col : 1   Pos : 281   Unix (LF)   UTF-8   INS

28.4 ReactJS Concepts

HTML in JSX

- To write HTML on multiple lines, put HTML inside parentheses
- HTML code must be wrapped in **ONE** top level element
- To write two sibling paragraphs:
  - Must put them inside parent element, like div element
- Alternatively, can use a "fragment" to wrap multiple lines:
  - Prevents unnecessarily adding extra nodes to DOM
  - Fragment looks like an empty HTML tag: <></>
- All html elements must be closed, even void ones
- JSX follows XML rules, and therefore HTML elements must be properly closed
- For void or self closing elements, use forward slash before ending angle bracket

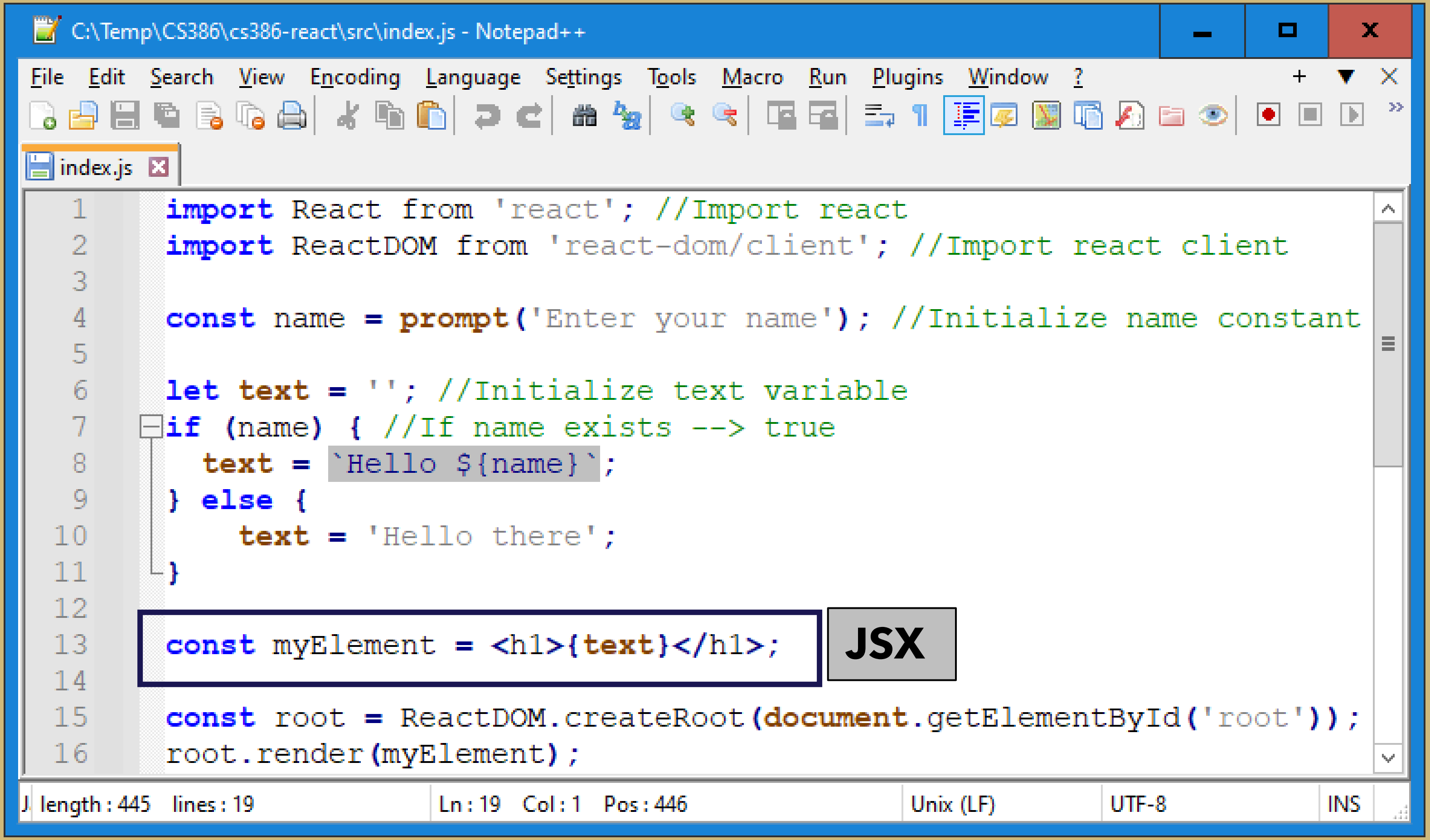
28.4 ReactJS Concepts

HTML in JSX

- class attribute is much used attribute in HTML
- JSX is rendered as JavaScript, class is reserved word in JavaScript
- Not allowed to use it in JSX
- Use attribute className instead in JavaScript/JSX
- React supports if statements, but not inside JSX
- To use conditional statements in JSX:
  - Place if statements outside of JSX
  - Or use ternary expression instead inside curly braces

28.4 ReactJS Concepts

Example 28-5:

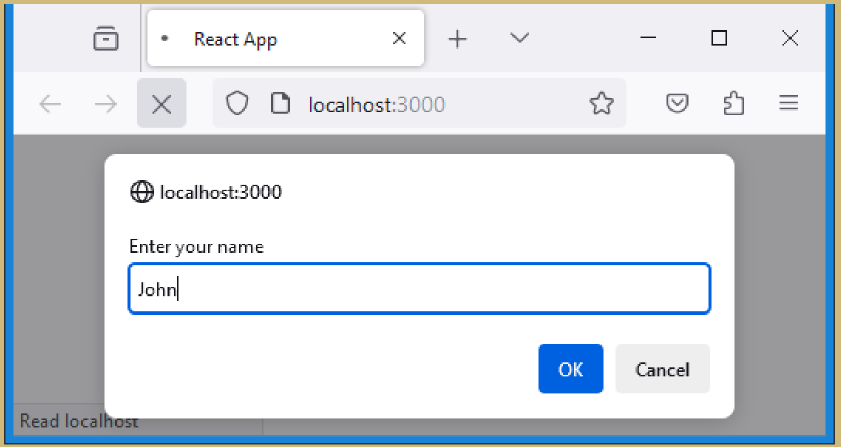


C:\Temp\CS386\cs386-react\src\index.js - Notepad++

```
import React from 'react'; //Import react
import ReactDOM from 'react-dom/client'; //Import react client

const name = prompt('Enter your name'); //Initialize name constant

let text = ''; //Initialize text variable
if (name) { //If name exists --> true
  text = `Hello ${name}`;
} else {
    text = 'Hello there';
}

const myElement = <h1>{text}</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

JSX

```
length : 445    lines : 19    Ln : 19    Col : 1    Pos : 446    Unix (LF)    UTF-8    INS
```

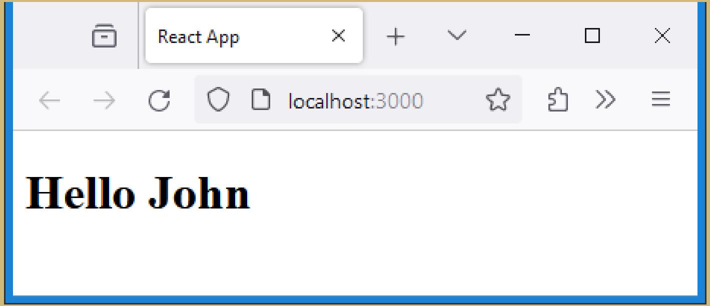


localhost:3000

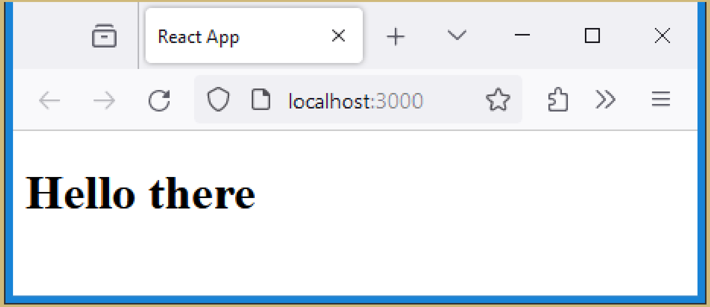
Enter your name

John

OK   Cancel



Hello John

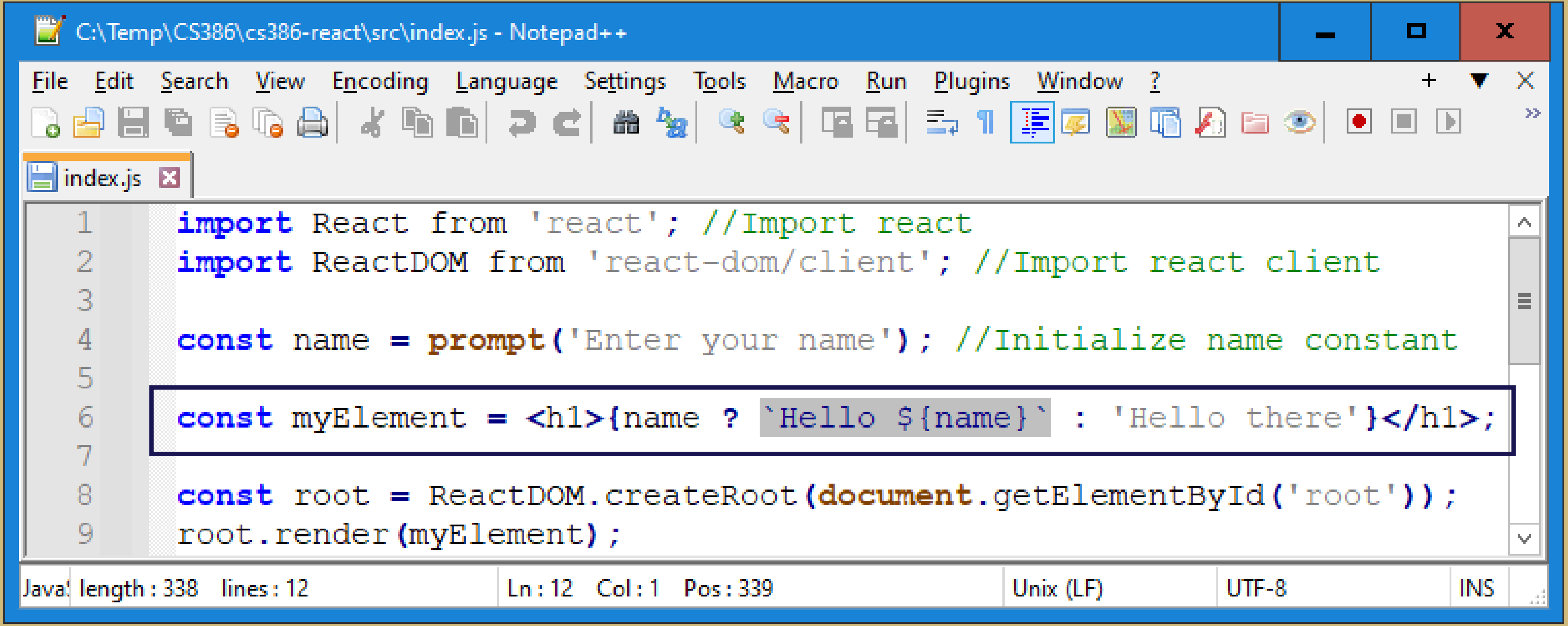


Hello there

28.4 ReactJS Concepts

Example 28-5:

- Using ternary operator


C:\Temp\CS386\cs386-react\src\index.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
index.js

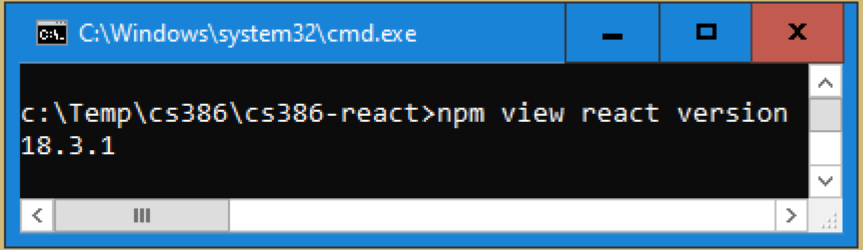
```
import React from 'react'; //Import react
import ReactDOM from 'react-dom/client'; //Import react client

const name = prompt('Enter your name'); //Initialize name constant

const myElement = <h1>{name ? `Hello ${name}` : 'Hello there'}</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

Java! length : 338 lines : 12
Ln : 12 Col : 1 Pos : 339
Unix (LF)
UTF-8
INS

28.4 ReactJS Concepts

React Components

- Components are independent and reusable bits of code
- Serve same purpose as JavaScript functions, but work in isolation and return HTML
- Components come in two types:
  - Class components
  - Function components
- In older React code bases, may find Class components primarily used
- It is now suggested to use Function components along with Hooks
- Hooks were added in React 16.8



```
c:\Temp\cs386\cs386-react>npm view react version
18.3.1
```

28.4 ReactJS Concepts

React Components

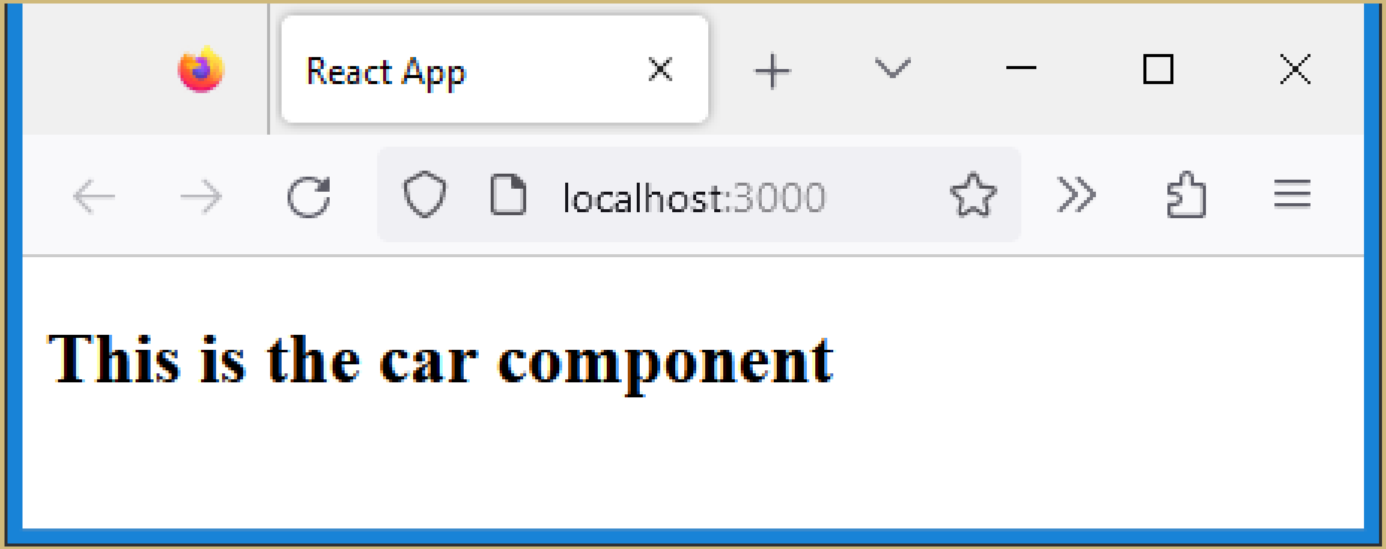
- When creating React components:
  - Component's name **MUST** start with upper case letter
- Function component returns HTML
- Function components can be written using much less code:
  - Compared to class method, are also easier to understand

Syntax:
```
function Component_name() {
    return (
        html
    )
}
//Render component
root.render(<Component_name />)
```

28.4 ReactJS Concepts

Example 28-6:

- ➢ Building first component
- ➢ Create function Car
  - ❑ Return h1 element with content as shown below:
- ➢ Use render method on root passing car component using html/xml syntax

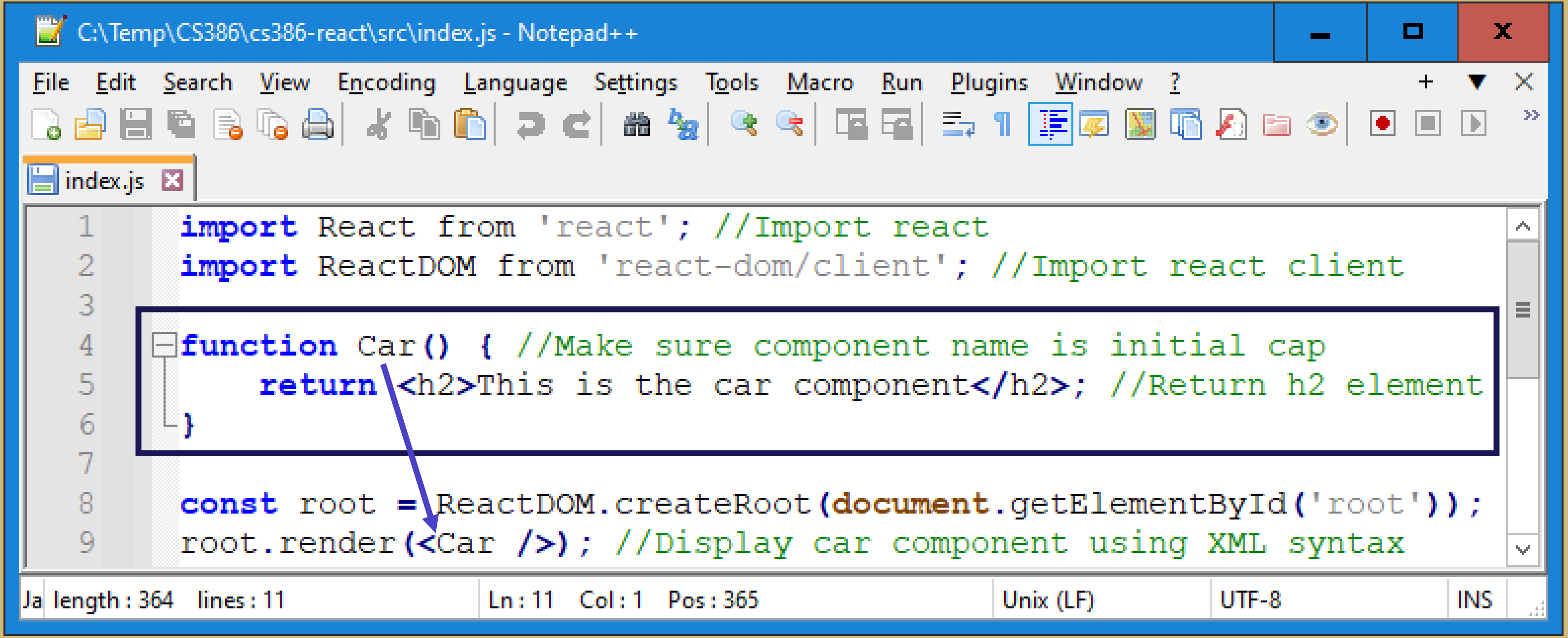


This is the car component

28.4 ReactJS Concepts

Example 28-6:

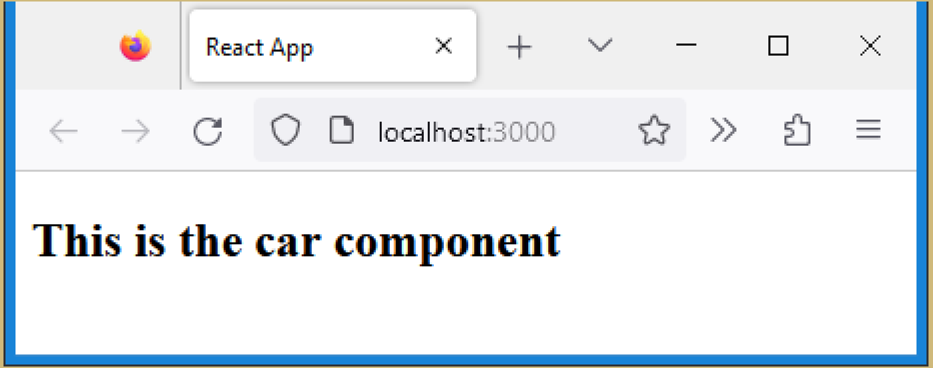
- Building first component
- Notice the html element syntax for displaying car element:
  - ❑ <Car />



C:\Temp\CS386\cs386-react\src\index.js - Notepad++

```
import React from 'react'; //Import react
import ReactDOM from 'react-dom/client'; //Import react client

function Car() { //Make sure component name is initial cap
    return <h2>This is the car component</h2>; //Return h2 element
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />); //Display car component using XML syntax
```

```
Ja length : 364    lines : 11    Ln : 11    Col : 1    Pos : 365    Unix (LF)    UTF-8    INS
```



React App

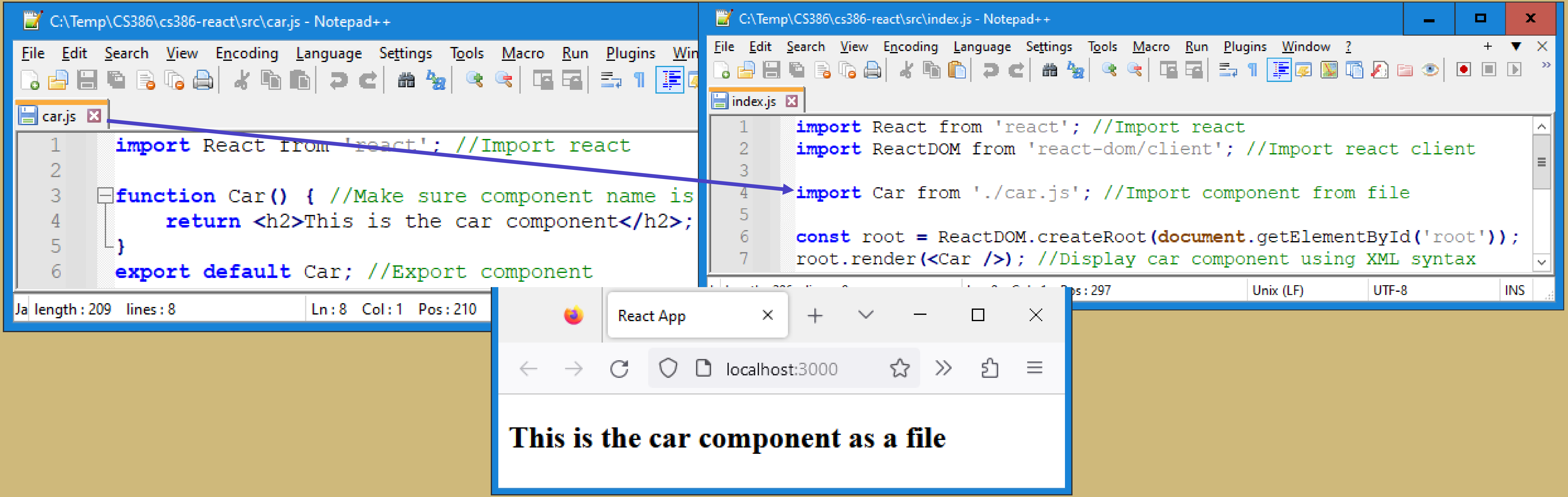
localhost:3000

This is the car component

28.4 ReactJS Concepts

Example 28-6:

- ➢ Place component in separate file (place file in src directory)
- ➢ If only one export, use export default component (easier to import)



C:\Temp\CS386\cs386-react\src\car.js - Notepad++

```
import React from 'react'; //Import react

function Car() { //Make sure component name is
    return <h2>This is the car component</h2>;
}
export default Car; //Export component
```

Ja length : 209 lines : 8 Ln : 8 Col : 1 Pos : 210

C:\Temp\CS386\cs386-react\src\index.js - Notepad++

```
import React from 'react'; //Import react
import ReactDOM from 'react-dom/client'; //Import react client

import Car from './car.js'; //Import component from file

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />); //Display car component using XML syntax
```

React App — localhost:3000

This is the car component as a file

28.4 ReactJS Concepts

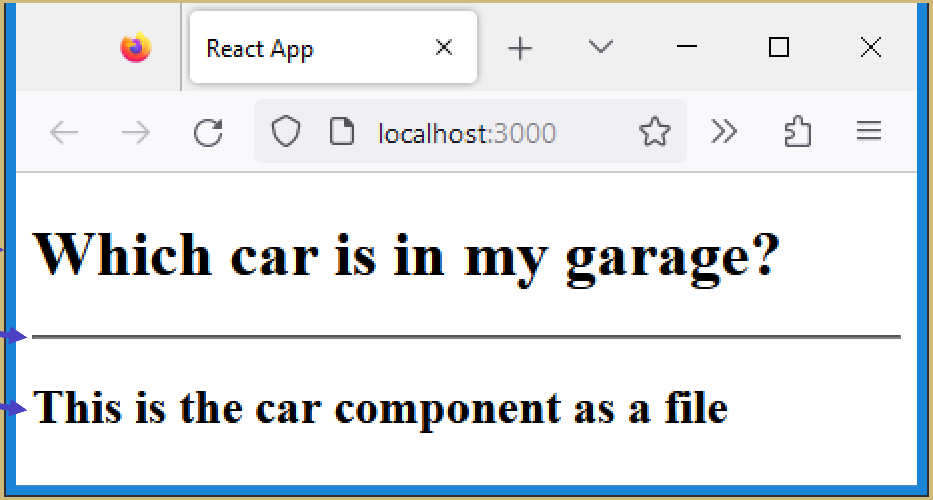
React Components

- Nested components, to be truly modular
- Next example shows two components, nested:
  - Garage and Car
- Load Car component into Garage
- Load Garage component into main file (index.js)

28.4 ReactJS Concepts

Example 28-7:

- Use file garage.js file in src folder
- Import:
  - Import 'react' into React
  - Import car component
- Create garage component using function:
  - Use empty element to wrap component
  - H1 header
  - Horizontal ruler
  - Display car component
- Default export Garage



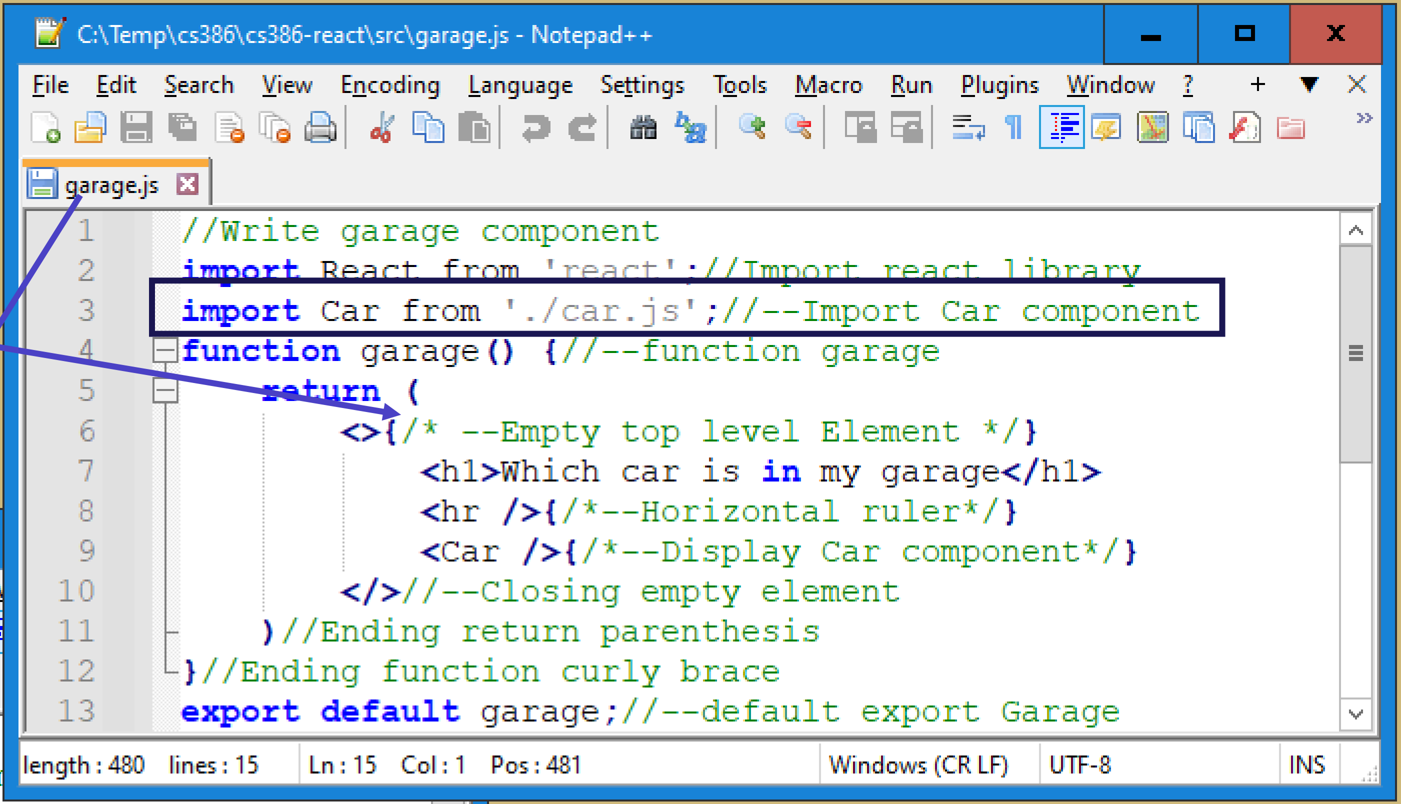
Which car is in my garage?

This is the car component as a file

28.4 ReactJS Concepts

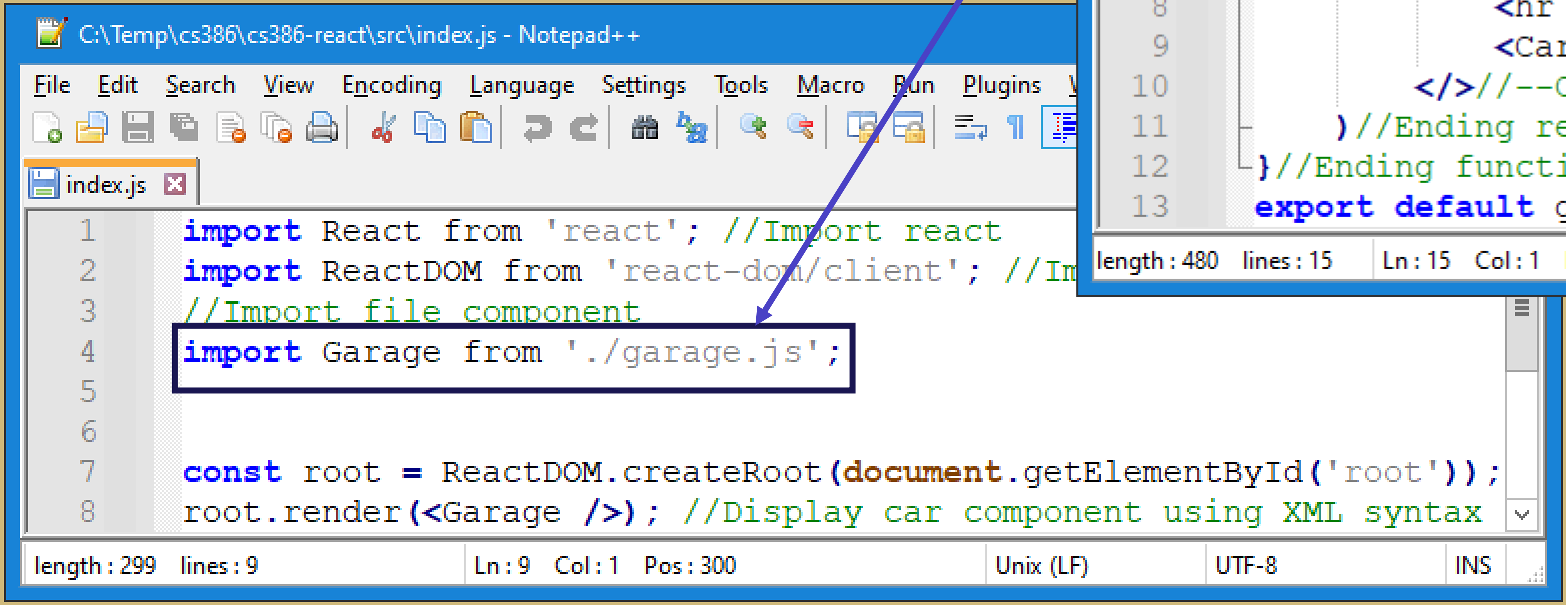
Example 28-7:

- Notice comments in JSX:
  - *{/* comment */}*
- Display Garage in main file, index.js


C:\Temp\cs386\cs386-react\src\garage.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
garage.js

```
//Write garage component
import React from 'react';//Import react library
import Car from './car.js';//--Import Car component
function garage() {//--function garage
    return (
        <>{/* --Empty top level Element */}
            <h1>Which car is in my garage</h1>
            <hr />{/*--Horizontal ruler*/}
            <Car />{/*--Display Car component*/}
        </>//--Closing empty element
    )//Ending return parenthesis
}//Ending function curly brace
export default garage;//--default export Garage
```

length : 480 lines : 15 Ln : 15 Col : 1 Pos : 481 Windows (CR LF) UTF-8 INS


C:\Temp\cs386\cs386-react\src\index.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins
index.js

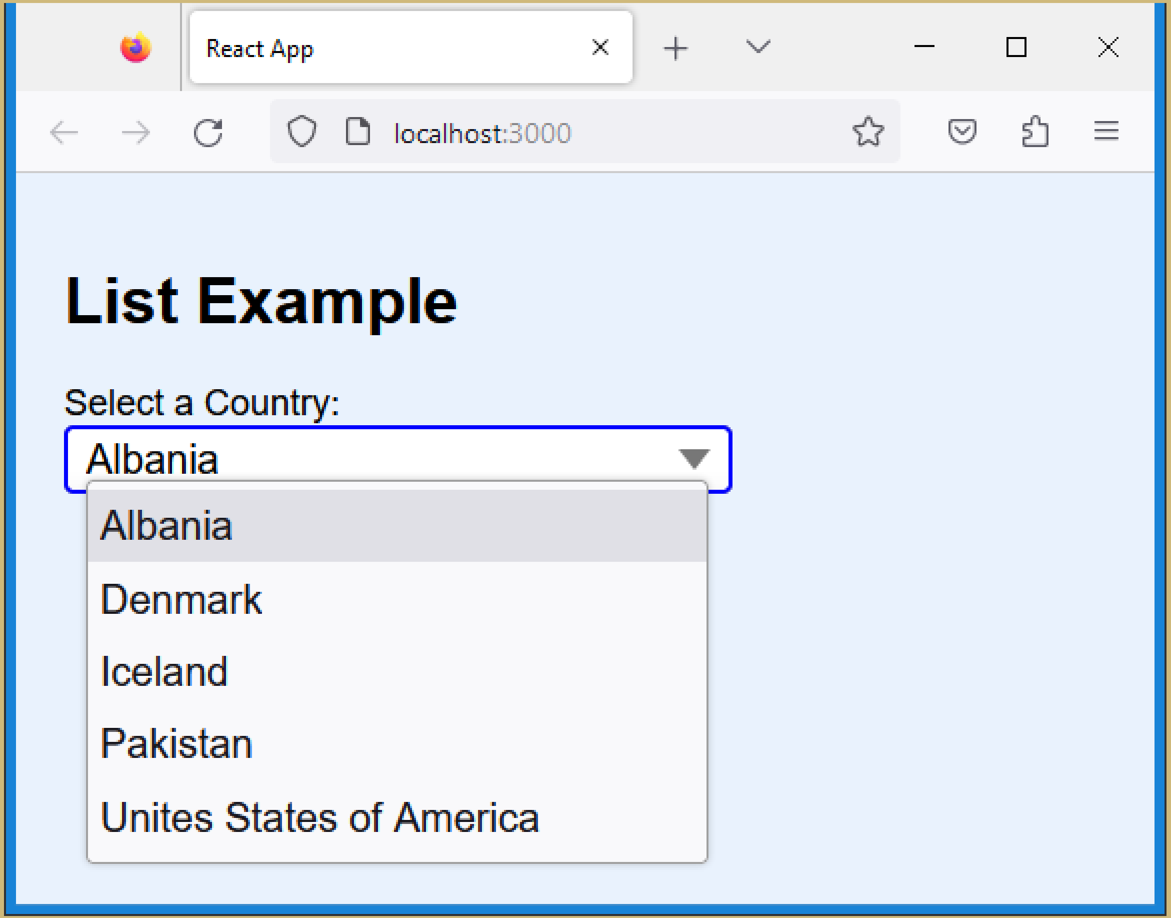
```
import React from 'react'; //Import react
import ReactDOM from 'react-dom/client'; //Im
//Import file component
import Garage from './garage.js';


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage />); //Display car component using XML syntax
```

length : 299 lines : 9 Ln : 9 Col : 1 Pos : 300 Unix (LF) UTF-8 INS

28.4 ReactJS Concepts

Example 28-8:

- Advanced example of a <select> component including CSS
- list.js:
  - Countries in array
  - Use map() to loop over array to assemble <options> markup
- list.css:
  - Regular stylesheet file
  - Import into list.js



List Example

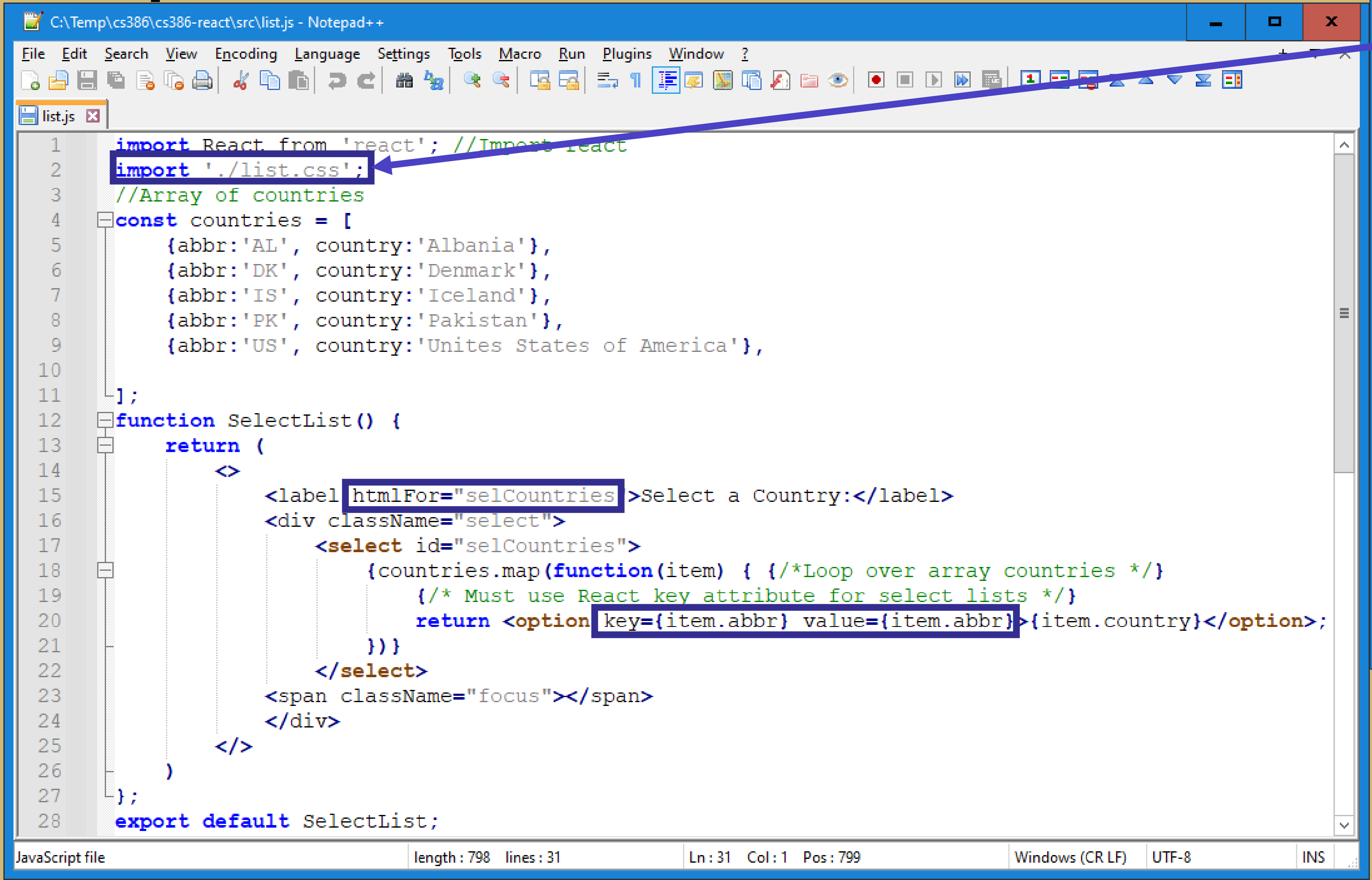
Select a Country:

Albania

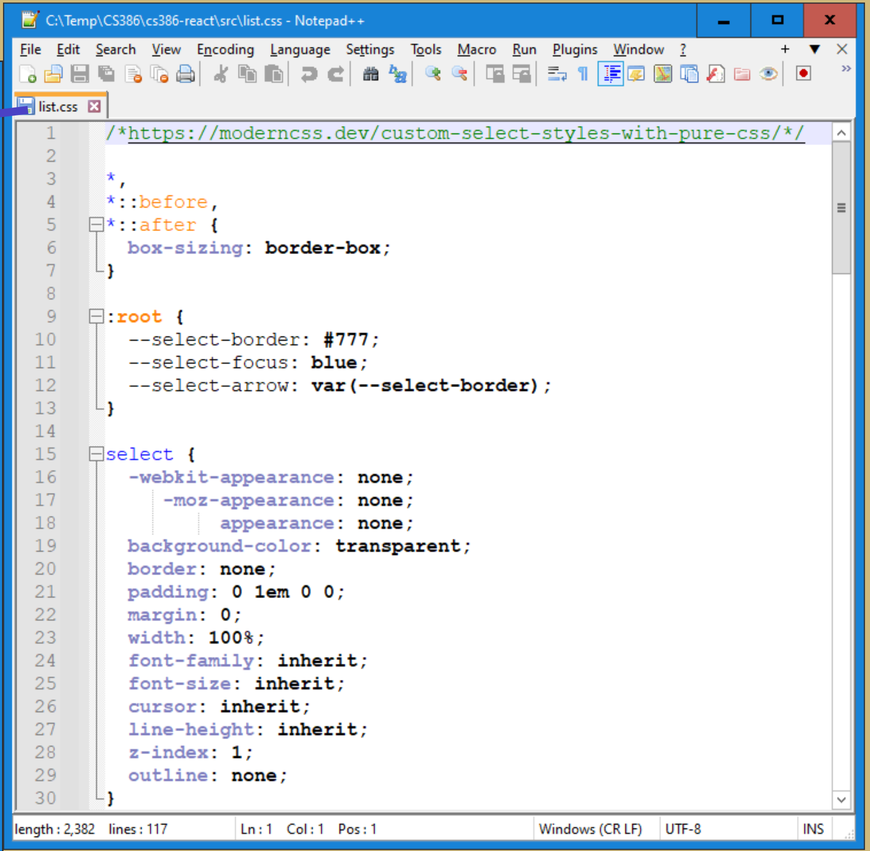
Albania
Denmark
Iceland
Pakistan
Unites States of America

28.4 ReactJS Concepts

Example 28-8:


C:\Temp\cs386\cs386-react\src\list.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
list.js
JavaScript file
length : 798 lines : 31
Ln : 31 Col : 1 Pos : 799
Windows (CR LF) UTF-8 INS

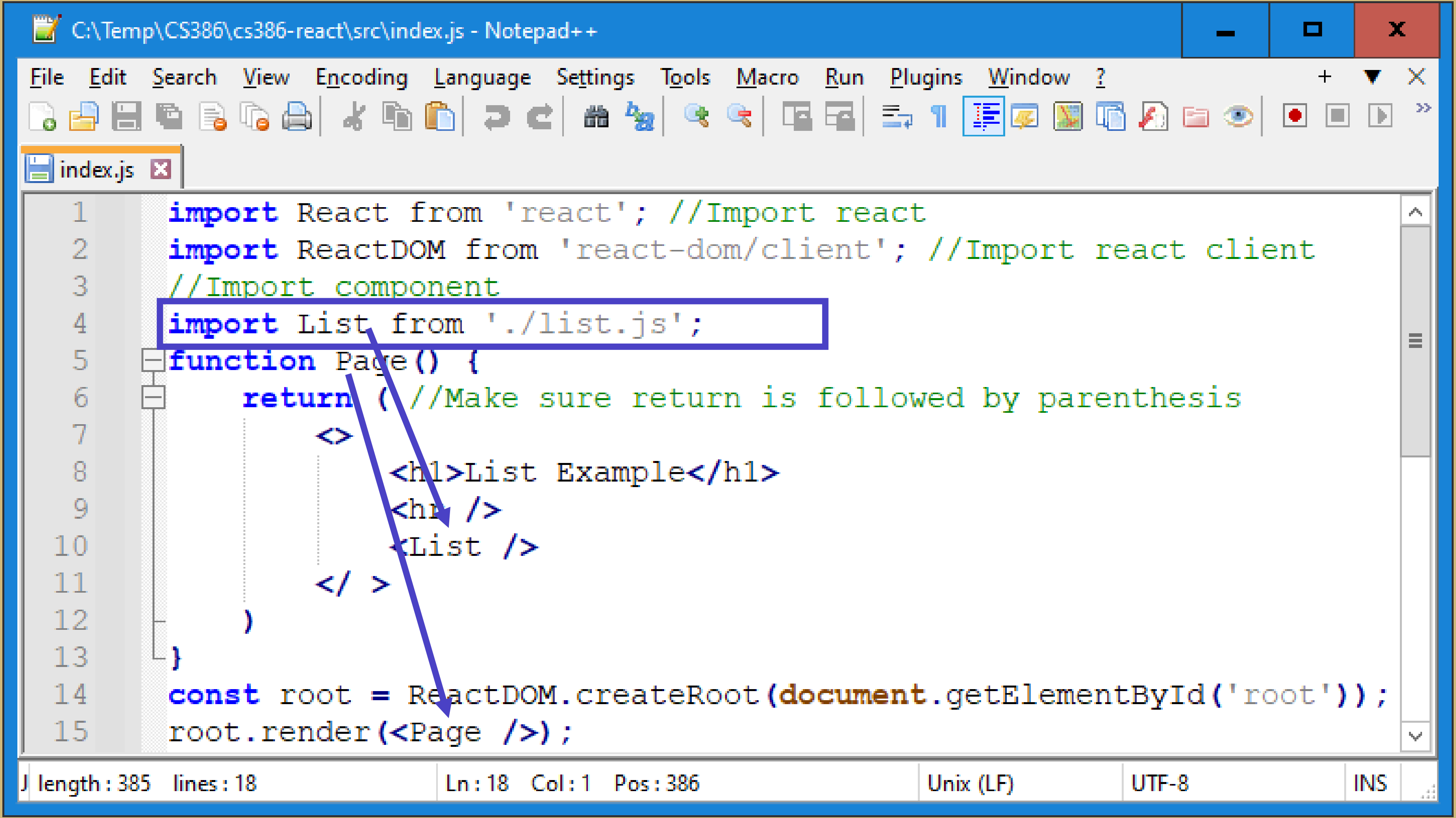
```
import React from 'react'; //Import react
import './list.css';
//Array of countries
const countries = [
     {abbr:'AL', country:'Albania'},
     {abbr:'DK', country:'Denmark'},
     {abbr:'IS', country:'Iceland'},
     {abbr:'PK', country:'Pakistan'},
     {abbr:'US', country:'Unites States of America'},

];
function SelectList() {
     return (
          <>
               <label htmlFor="selCountries">Select a Country:</label>
               <div className="select">
                    <select id="selCountries">
                         {countries.map(function(item) { {/*Loop over array countries */}
                              {/* Must use React key attribute for select lists */}
                              return <option key={item.abbr} value={item.abbr}>{item.country}</option>;
                         })}
                    </select>
               <span className="focus"></span>
               </div>
          </>
     )
};
export default SelectList;
```


C:\Temp\CS386\cs386-react\src\list.css - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
list.css
length : 2,382 lines : 117
Ln : 1 Col : 1 Pos : 1
Windows (CR LF) UTF-8 INS

```
/*https://moderncss.dev/custom-select-styles-with-pure-css/*/

*,
*::before,
*::after {
  box-sizing: border-box;
}

:root {
  --select-border: #777;
  --select-focus: blue;
  --select-arrow: var(--select-border);
}

select {
  -webkit-appearance: none;
     -moz-appearance: none;
          appearance: none;
  background-color: transparent;
  border: none;
  padding: 0 1em 0 0;
  margin: 0;
  width: 100%;
  font-family: inherit;
  font-size: inherit;
  cursor: inherit;
  line-height: inherit;
  z-index: 1;
  outline: none;
}
```

28.4 ReactJS Concepts

Example 28-8:


C:\Temp\CS386\cs386-react\src\index.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
index.js
length : 385 lines : 18
Ln : 18 Col : 1 Pos : 386
Unix (LF)
UTF-8
INS

```
import React from 'react'; //Import react
import ReactDOM from 'react-dom/client'; //Import react client
//Import component
import List from './list.js';
function Page() {
    return ( //Make sure return is followed by parenthesis
        <>
            <h1>List Example</h1>
            <hr />
            <List />
        </ >
    )
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Page />);
```