JSX

React

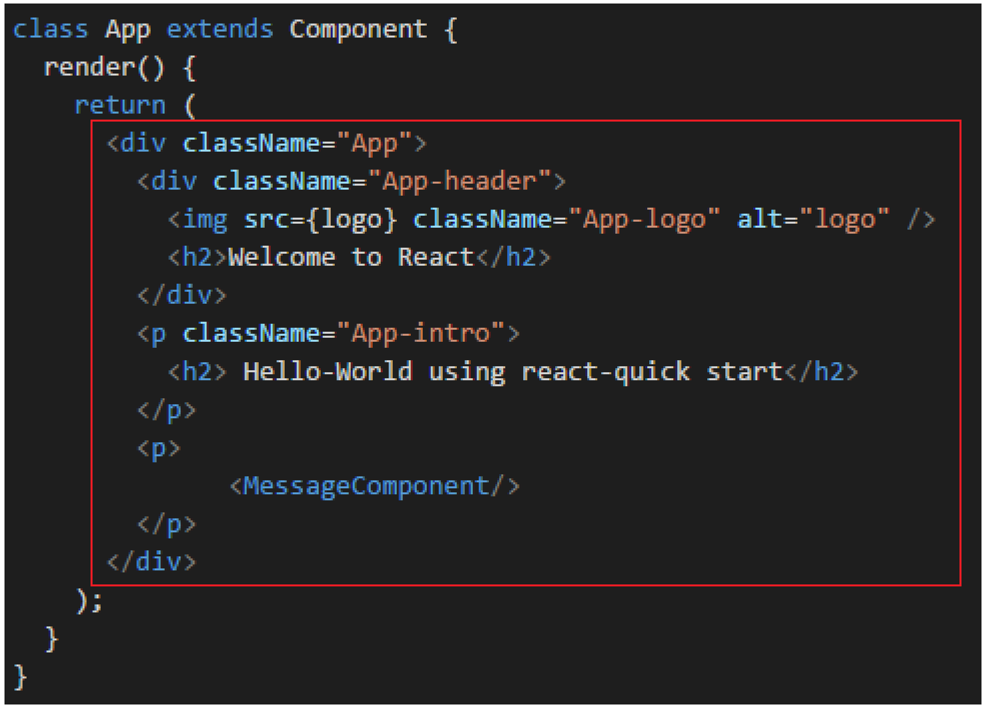
Presented by: Pankaj Sharma

Topics to be covered

- JSX Introduction
- Pre- requisites
- JSX Compiler
- JSX Syntax
- JSX Examples
- Precompiling JSX

JSX Introduction

JSX allows us xml like opening and closing tags to define HTML elements.



```
class App extends Component {
  render() {
    return (
      <div className="App">
        <div className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h2>Welcome to React</h2>
        </div>
        <p className="App-intro">
          <h2> Hello-World using react-quick start</h2>
        </p>
        <p>
              <MessageComponent/>
        </p>
      </div>
    );
  }
}
```

Code inside red square is an example of JSX

JSX Prerequisite

- Web server – this is to use extensions such as react-detector as they require the http protocol.
- You need to be familiar with HTML, CSS, JavaScript
- You need to be familiar with ReactJS basics.
- Text editor or IDE that support HTML, CSS, JS
- JavaScript enabled browser.
- Command line tool for node commands.

JSX Compiler

- JSX compiler is used to convert the JSX into JavaScript code.
- Here we will be using Babel for testing and learning purpose.
- Babel is slow and not recommended in production environment.
- For production environment, Node architecture is the recommended approach.

JSX Syntax

We have already written the JSX in our React Overview. Let us rewrite the code and understand the JSX, and different JSX expressions. Let us create basic template for our demos.

```
Day2/jsxdemo_01.html
---------------------------------------------------
<!DOCTYPE html>
<html lang="en">
<head>
<title>JSX Demo</title>
</head>
<body>
<h1>JSX- Demo 01</h1><hr>
<div id="target"> <!-- here ReactJS code will be rendered --> </div>
<script src="react.js"></script> <script src="react-dom.js"></script> <script
src="babel.min.js"></script>
    <script type="text/babel">
/* Create React Component */
var HelloComponent=React.createClass({
render : function(){ return <h1>Hello</h1>; } });
/* Render an instance of HelloComponent into Document body */
ReactDOM.render(<HelloComponent/>,document.getElementById('target'));
</script>
</body>
</html>
```

<h1>Hello</h1> expression in HelloComponent is JSX.

JSX Syntax

var HelloComponent - defines a variable that is used to store the React Component
React.createClass({…}); - instantiates a react component
render :function (){….} –this anonymous function is used to output the HTML content in the browser.
return (html component) – contains the html contents that we want to output in the web browser.

JSX Examples

- **Nested Elements**
- **JavaScript**
- **Style**
- **Comments**
- **Function Creation and calling**
- **Practical Assignments**

JSX Examples – Nested Elements

Let us set up the actual React Environment, on that we will experiment all JSX examples.

1: Create a folder in any drive (d:/react_demos)
2: Open the command prompt and go inside the react_demo folder
3: Write below command to create project for jsxdemos

```
create-react-app jsxdemos
```

Note : you must have node installed and **npm install -g create-react-app** should have been run before create-react-app command. Wait for the process.

4: Go inside the jsxdemos and run **npm start.** Your application will run with initial configuration.
5: Remove below files.

* src/App.css
* src/App.js
* src/ App.test.js
* src/index.js (remove the code, do not delete the file.)

6: Add code inside the **index.js.** That is shown on next slide.

JSX Examples – Nested Elements

```
day-2/jsxdemos/src/index.js
-------------------------------------------------
import React from 'react';
import ReactDOM from 'react-dom';

class HelloMessage extends React.Component{
render(){
return (
<h1>Hello Message</h1>
)
}
}

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

Assignment : Add below code and see the changes.

```
return (
        <h1>Main Heading</h1>
        <h1>Sub Heading</h1>
        <p1>This is paragraph!</p1>
)
```

JSX Examples – Nested Elements : Solution

```
day-2/jsxdemos/src/index.js
-------------------------------------------------
import React from 'react';
import ReactDOM from 'react-dom';

class HelloMessage extends React.Component{
render(){
return (
<div>
<h1>Main Heading</h1>
<h2>Sub Heading</h2>
<p>This is paragraph!</p>
</div>
)
}
}

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

We need one parent container to enclose all the child tags.
Now check the output on browser.

JSX Examples – JavaScript

JavaScript expressions can be used inside JSX. You just need to wrap it with curly brackets {}. You can add any javascript code in {}.

```
day-2/jsxdemos/src/index.js
--------------------------------------------------
import React from 'react';
import ReactDOM from 'react-dom';

class HelloMessage extends React.Component{
render(){
var message='This is paragraph!';
var count=0;
return (
<div>
<h1>Main Heading</h1>
<h2>Sub Heading</h2>
<p>{message}</p>
<p>This is the counter value : {count+1}</p>
{alert('Hi')}
</div>
);
}
}

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

JSX Examples – JavaScript

You can use ternary operator in JSX instead of if else logic, if else, for, while are not allowed in JSX.

```
day-2/jsxdemos/src/index.js
--------------------------------------------------
import React from 'react';
import ReactDOM from 'react-dom';

class HelloMessage extends React.Component{
render(){
var message='This is paragraph!';
var count=1;
return (
<div>
<h1>Main Heading</h1>
<h2>Sub Heading</h2>
<p>{message}</p>
<h3>{count===0? 'No one visited site':'People Started visiting our site'}</h3>
<p>This is the counter value : {count+1}</p>
</div>
);
}
}

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

JSX Examples – Inline Style

JSX also allows to add inline style.

```
day-2/jsxdemos/src/index.js
--------------------------------------------------
import React from 'react';
import ReactDOM from 'react-dom';

class HelloMessage extends React.Component{
render(){
var message='This is paragraph!';
var count=1;
var myStyle={
color:'red'
}
return (
<div>
<h1 style={myStyle}>Main Heading</h1>
<h2>Sub Heading</h2>
<p>{message}</p>
<h3>{count===0? 'No one visited site':'People Started visiting our site'}</h3>
<p>This is the counter value : {count+1}</p>
</div>
);
}
}

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

JSX Examples – Adding External style

JSX also allows to add external CSS. Create below code in index.js file.

```
day-2/jsxdemos/src/index.js
--------------------------------------------------
import ReactDOM from 'react-dom';
-----------
return (
<div>
<h1 style={myStyle}>User Registration</h1>
<div>
<table>
<tr>
<td>Name</td>
<td><input type="text" name="name"/></td>
</tr>
<tr>
<td>Contact</td>
<td><input type="text" name="contact"/></td>
</tr>
<tr>
<td className='rightAlign' colSpan="2"><button>Register</button></td>
</tr>
</table>
</div>
</div>
);
}
--------
```

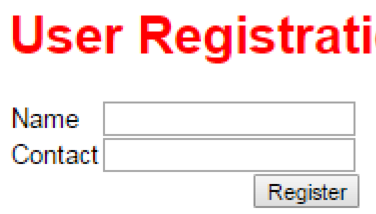
Note : All the form tags should be self enclosed or either with closing tags. It is the JSX convention.

JSX Examples – Adding External style

Create below css in index.css file, save the changes and check your browser.

```
day-2/jsxdemos/src/index.css
--------------------------------------------------
.rightAlign{
text-align: right;
}
```

You will be getting the output as below.


User Registration

JSX Examples – Function Creation and calling.

We had hard coded the user Registration, this can also be converted in component. Create getFormTile() method in your HelloMessge Component, and call that in {} at appropriate position.

```
day-2/jsxdemos/src/index.css
--------------------------------------------------
class HelloMessage extends React.Component{
getFormTile(){
return "User Registration Form";
}
------------
return (
<div>
<h1 style={myStyle}>{this.getFormTile()}</h1>
------------
);
}
}
```

JSX Examples – Comments

We can write the comments in {}

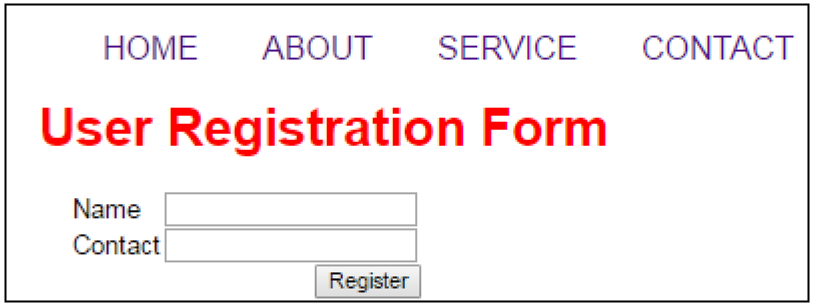
```
day-2/jsxdemos/src/index.js
---------------------------------------------------
return (
<div>
{/*
applying the inline style
multiple line comment can be added here
*/
}
<h1 style={myStyle}>Main Heading</h1>
<h2>Sub Heading</h2>
<p>{message}</p>
<h3>{count===0? 'No one visited site':'People Started visiting our site'}</h3>
<p>This is the counter value : {count+1}</p>
</div>
);
```

JSX Examples – Practical Assignment

# Create a user registration page, that should have main menu with navigation options. Form title should be customizable.

Note :

- There should be separate component for navigation.
- Form title component should be separate
- Form component should be separate.
- Your output should be as below.



Pre Compiling JSX - Introduction

JSX makes it easier to create complex user interfaces but compiling it at runtime in production slows down the performance of our React Application. Precompiling JSX into JavaScript before deployment solves this problem.

We will use babel command line utility to compile our JSX code into JavaScript Code. We will use NodeJS Package Manager (NPM) to install babel cli.

Pre Compiling JSX

To understand this section we have some prerequisites:

- NodeJs
- Tomcat server 7.0 up and running using command prompt
- Any good IDE for code writing

Objective:
We would like to write the JSX code and want to compile it into .js code. Any time any changes should automatically reflect in .js file.
There will be one .html file, there we will be including .js file.
We will be writing code in React JS or in JSX specifically, and will not touch the .js file, that code will be automatically written.

Pre Compiling JSX

Steps
1: Create a folder in any location and name it as ***jsxprecompilationdemo***
2: Create a new directory ***src*** in ***jsxprecompilationdemo*** folder
3: Create ***hello.jsx*** in ***src*** directory
4: add the below code in ***hello.jsx*** file

```
Jsxprecompilationdemo/src/hello.jsx
--------------------------------------------------
var Hello=React.createClass({
render:function(){
return(
<div>
<h3>!!!JSX Precompilation Example!!!</h3>
<p>This example is created mannually, for showing the precompilation process of JSX.</p>
</div>
);
}
});

ReactDOM.render(<Hello/>,document.getElementById('target'));
```

Let us now precompile the ***hello.jsx*** file
5: Create ***lib*** folder in ***jsxprecompilationdemo*** folder

Pre Compiling JSX

Steps

6: Create a new file ***.babelrc*** in the root directory or ***jsxprecompilationdemo***

7: Add the below code in ***.babelrc*** file

```
Jsxprecompilationdemo/.babelrc
----------------------------------------------------
{
"presets": ['react']
}
```

8: Copy ***jsxprecompilationdemo*** folder and paste in : ***C:\apache-tomcat-7.0.64\webapps***

9: Open ***jsxprecompilationdemo*** in visual studio, which you have pasted in ***C:\apache-tomcat-7.0.64\webapps***

10: Open command prompt and copy the path till : ***C:\apache-tomcat-7.0.64\webapps\jsxprecompilationdemo***

11: change the directory in command prompt by writing below command.

cd C:\apache-tomcat-7.0.64\webapps\jsxprecompilationdemo :--- hit enter

12: now write below commands

npm install -g babel-cli

npm install babel-preset-es2015 babel-preset-react

Pre Compiling JSX

Steps

13: write the below command as well.

```
babel --presets es2015,react --watch src/ --out-dir lib/
```

This command will read any .jsx file from src and compile it in .js and put it in lib folder.

Any changes in .jsx will be automatically reflected in .js file.

After running the above command you will be able to get the hello.js file in lib folder, and find the below code.

Pre Compiling JSX

```
Jsxprecompilationdemo/lib/hello.js
--------------------------------------------------
'use strict';

var Hello = React.createClass({
displayName: 'Hello',

render: function render() {
return React.createElement(
'div',
null,
React.createElement(
'h3',
null,
'!!!JSX Precompilation Example!!!'
),
React.createElement(
'p',
null,
'This example is created mannually, for showing the precompilation process of JSX.'
)
);
}
});

ReactDOM.render(React.createElement(Hello, null), document.getElementById('target'));
```

Pre Compiling JSX

Steps

14: now add ***react.js*** and ***react-dom.js*** in root folder ***jsxprecompilationdemo***

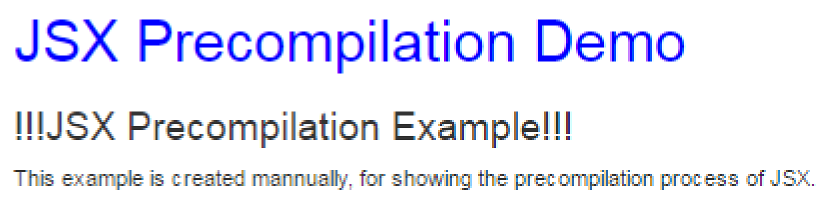
15: Create ***jsxdemo.html*** file in the root of ***jsxprecompilationdemo***

16: write below code in ***jsxdemo.html***

```
Jsxprecompilationdemo/jsxdemo.html
-------------------------------------------------------
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>JSX Precompilation Demo</title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body>
<div class="container">
<h1 style="color:blue">JSX Precompilation Demo</h1>
<div id="target">
<!-- hello component will be rendered here -->
</div>
</div>
<script src="react.js"></script>
<script src="react-dom.js"></script>
<script src="lib/hello.js"></script>
</body>
</html>
```

Pre Compiling JSX

Steps
17: now run the tomcat server by clicking on startup.bat file.
18: once the server is up and running, open the browser and type below address in browser address bar.
http://localhost:8080/jsxprecompilationdemo/jsxdemo.html
You will be getting the below output.


JSX Precompilation Demo

!!!JSX Precompilation Example!!!

This example is created mannually, for showing the precompilation process of JSX.

Now try to make any changes in your hello.jsx file, and let command prompt be open for node.
Observe that any changes you made in .jsx file, changes are also reflected in .js file. You do not need to recompile.
Because you have run babel jsx compilation in watch mode.

Thank You!

Email: info@yash.com
Web: www.yash.com

Presented by: Pankaj Sharma