**ReactJS Tutorial Part #3: ReactJSX Functions -** [Teman Ngoding](https://temanngoding.medium.com/?source=post_page-----ad2457f6bd76--------------------------------) Apr 26, 2022

**React JSX**

How are you all friends, I hope you are all in good health…. Amen….

This time we will discuss React JSX, All React components have a render function. The render function determines the HTML output of the React component. JSX (JavaScript Extension), is a React extension that allows writing JavaScript code that looks like HTML. In other words, JSX is the HTML-like syntax used by React that extends ECMAScript so that HTML-like syntax can coexist with JavaScript/React code. The syntax is used by the preprocessor (that is, a transpiler such as babel) to convert HTML-like syntax into standard JavaScript objects to be parsed by the JavaScript engine.

JSX provides you to write HTML/XML-like structures (for example, tree structures like DOM) in the same file where you write JavaScript code, then the preprocessor converts these expressions into actual JavaScript code. Just like XML/HTML, JSX tags have tag names, attributes, and children.

**Sample case**

Here, we will write the JSX syntax in a JSX file and see the appropriate JavaScript code that the preprocessor converts.

**JSX File**

<div>Hello JavaTpoint</div>

**Corresponding Output**

React.createElement("div", null, "Hello Teman Ngoding");

So we can see the syntax above, the first is the element with <div> and the second is the attribute passed in the div tag.

**Why Use JSX**

* It is faster than plain JavaScript because it performs optimizations when translating code to JavaScript.
* Can put markup and logic in separate files, React uses components that contain both. We will study the components in the next section.
* Can easily find errors at compile time.
* It makes it easier to create templates.
* For all of you who want to use more than one element, you need to wrap it in one container element. Here, we are using a div as a container element which has three nested elements in it.

For those of you who want to use more than one element, you need to wrap it in a single container element. Here, we are using a div as a container element which has three nested elements in it.

import React, { Component } from 'react';   
*class* App extends Component{   
 render(){   
 return(   
 <div>   
 <h1>Teman Ngoding</h1>   
 <h2>Ngoding Together</h2>   
 </div>   
 );   
 }   
}   
export default App;

Output:

Graphical user interface, text, application, chat or text message

Description automatically generated

**JSX attribute**

If you often use HTML, then JSX attribute JSX is the same as regular HTML. We can also use our own custom attributes in JSX. For custom attributes, we need to use the data- prefix. In the example below, we have used the special attribute data-demoAttribute as the attribute for the tag.

Example:

import React, { Component } from 'react';   
*class* App extends Component{   
 render(){   
 return(   
 <div>   
 <h1>Teman Ngoding</h1>   
 <h2>Ngoding Together</h2>   
 </div>   
 );   
 }   
}   
export default App;

In JSX, we can define attribute values in two ways:

1. String Literal: We can specify the attribute value in double quotes:

var element = <h2 className = "firstAttribute">Hello JavaTpoint</h2>;

Example:

import React, { Component } from 'react';   
class App extends Component{   
 render(){   
 return(   
 <div>   
 <h1 className = "hello" >Teman Ngoding</h1>   
 <p data-demoAttribute = "demo">Belajar bersama teman ngoding sangat menyenangkan.</p>   
 </div>   
 );   
 }   
}   
export default App;

Output:

Graphical user interface

Description automatically generated with medium confidence

2. We can assign attribute values as expressions using curly braces { }:

var element = <h2 className = {varName}>Hello JavaTpoint</h2>;

Example:

import React, { Component } from 'react';   
class App extends Component{   
 render(){   
 return(   
 <div>   
 <h1 className = "hello" >{25+20}</h1>   
 </div>   
 );   
 }   
}   
export default App;

Output:

Graphical user interface

Description automatically generated with medium confidence

**JSX Comments**

JSX uses comments that start with /\* and end with \*/ and in curly braces {} just like JSX expressions. The example below shows how to use comments in JSX.

import React, { Component } from 'react';   
*class* App extends Component{   
 render(){   
 return(   
 <div>   
 <h1 className = "hello" >Hello Teman Ngoding</h1>   
 {/\* This is a comment in JSX \*/}   
 </div>   
 );   
 }   
}   
export default App;

**JSX Styling**

React always recommends to use inline styles. To set inline styles, you need to use camelCase syntax. React automatically allows appending px after the number value on specific elements. The following example shows how to use styling in the element.

import React, { Component } from 'react';   
*class* App extends Component{   
 render(){   
 var myStyle = {   
 fontSize: 80,   
 fontFamily: 'Courier',   
 color: '#003300'   
 }   
 return (   
 <div>   
 <h1 style = {myStyle}>www.temanngoding.com</h1>   
 </div>   
 );   
 }   
}   
export default App;

Output:

A picture containing logo

Description automatically generated

[ReactJS Tutorial Part #1: Introduction to ReactJS Basics](https://temanngoding.com/tutorial-reactjs-1-pengenalan-dasar-reactjs/)

[ReactJS Tutorial #2: ReactJS Directory Structure](https://temanngoding.com/tutorial-react-js-2-struktur-direktori-react-js/)