# ReactJS Tutorial

*Taken from https://reactjs.org/docs/introducing-jsx.html*

## Introducing JSX

This is an example of JSX

**const element = <h1>Hello, world!</h1>;**

The above tag is neither a string nor html.

JSX is a syntax extension to JavaScript.

JSX is used in React to describe what the UI should look like, and has full power of Javascript.

JSX produces React elements, which are then rendered in the DOM.

## Why is JSX used?

The following reasons, which are combined with rendering logic and UI logic, JSX is used:-

* Handling of events
* Changes in state
* Display of data

## Separation of Concerns

React uses units called ‘components’ to keep both mark-up and logic together, instead of in separate files (ie) putting mark-up in JS.

JSX isn’t required specifically in React but most people find it useful when working with UI inside JavaScript code. In addition, it also shows useful error messages.

## Embedding Expressions in JSX

Below we declare the variable ‘name’ and use it inside JSX by wrapping it in curly braces:

**const name = 'Josh Perez';const element = <h1>Hello, {name}</h1>;**

**ReactDOM.render(**

**element,**

**document.getElementById('root')**

**);**

**ReactDOM.render(**

**element,**

**document.getElementById('root')**

**);**

2

In the example below, we embed the result of calling a JavaScript function, formatName(user), into an <h1> element.

JavaScript expressions (such as 1 + 1, user.firstName, formatName(user) can also be put inside curly braces. In example below, the result of calling a Javascript function is embedded in curly braces.

**function formatName(user) {**

**return user.firstName + ' ' + user.lastName;**

**}**

**const user = {**

**firstName: 'John',**

**lastName: 'Doe'**

**};**

**const element = (**

**<h1>**

**Hello, {formatName(user)}! </h1>**

**);**

**ReactDOM.render(**

**element,**

**document.getElementById('root')**

**);**

1. Result: Hello, John Doe!

## JSX is an Expression Too

Once compiled, JSX expressions become regular JavaScript function calls and evaluate to JavaScript objects.

This means that you can use JSX inside of **if statements** and **for loops**, **assign** it to **variables**, **accept** it as arguments, and **return** it from functions:

**function getGreeting(user) {**

**if (user) {**

**return <h1>Hello, {formatName(user)}!</h1>; }**

**return <h1>Hello, Stranger.</h1>;}**