QUICK START

Getting Started
Tutorial

GUIDES

Why React?
Displaying Data
JSX in Depth
JSX Gotchas
Interactivity and Dynamic UIs
Multiple Components
Reusable Components
Forms
Working With the Browser
More About Refs
Tooling integration
Reference

JSX in Depth

JSX is a JavaScript XML syntax transform recommended for use with React.

Why JSX?

React works out of the box without JSX. Simply construct your markup using the functions on `React.DOM`. For example, here's how to construct a simple link:

Code

```
var link = React.DOM.a({href: 'http://facebook.github.io/react'}, 'React');
```

We recommend using JSX for many reasons:

- It's easier to visualize the structure of the DOM.
- Designers are more comfortable making changes.
- It's familiar for those who have used MXML or XAML.

The Transform

JSX transforms from an XML-like syntax into native JavaScript. XML elements and attributes are transformed into function calls and objects, respectively.

Code

```
var Nav;
// Input (JSX):
var app = <Nav color="blue" />;
// Output (JS):
var app = Nav({color:"blue"});
```

JSX also allows specifying children using XML syntax:

Code

```
var Nav, Profile;
// Input (JSX):
var app = <Nav color="blue"><Profile>click</Profile></Nav>;
// Output (JS):
var app = Nav({color:"blue"}, Profile(null, "click"));
```

Use the JSX Compiler to try out JSX and see how it desugars into native JavaScript.

If you want to use JSX, the Getting Started guide shows how to setup compilation.

> **Note:**
>
> Details about the code transform are given here to increase understanding, but your code should not rely on these implementation details.

React and JSX

React and JSX are independent technologies, but JSX was primarily built with React in mind. The two valid uses of JSX are:

- To construct instances of React DOM components (`React.DOM.*`).
- To construct instances of composite components created with `React.createClass()`.

React DOM Components

To construct a `<div>` is to create a variable that refers to `React.DOM.div`.

```
var app = <div className="appClass">Hello, React!</div>;
```

React Component Components

To construct an instance of a composite component, create a variable that references the class.

Code

```
var MyComponent = React.createClass({/*...*/});
var app = <MyComponent someProperty={true} />;
```

See Component Basics to learn more about components.

> **Note:**
>
> Since JSX is JavaScript, identifiers such as `class` and `for` are discouraged as XML attribute names. Instead, React DOM components expect attributes like `className` and `htmlFor`, respectively.

DOM Convenience

Having to define variables for every type of DOM element can get tedious (e.g. `var div, span, h1, h2, ...`). JSX provides a convenience to address this problem by allowing you to specify a variable in an `@jsx` docblock field. JSX will use that field to find DOM components.

Code

```
/**
 * @jsx React.DOM
 */
```

```
var tree = <Nav><Span /></Nav>;
// Output (JS):
var tree = Nav(null, React.DOM.span(null));
```

> **Remember:**
>
> JSX simply transforms elements into function calls and has no notion of the DOM. The docblock parameter is only a convenience to resolve the most commonly used elements. In general, JSX has no notion of the DOM.

JavaScript Expressions

Attribute Expressions

To use a JavaScript expression as an attribute value, wrap the expression in a pair of curly braces (`{}`) instead of quotes (`""`).

Code

```
// Input (JSX):
var person = <Person name={window.isLoggedIn ? window.name : ''} />;
// Output (JS):
var person = Person({name: window.isLoggedIn ? window.name : ''});
```

Child Expressions

Likewise, JavaScript expressions may be used to express children:

Code

```
// Input (JSX):
```

```
var content = Container(null, window.isLoggedIn ? Nav(null) : Login(null));
```

Comments

It's easy to add comments within your JSX; they're just JS expressions:

Code

```
var content = <Container>{/* this is a comment */}<Nav /></Container>;
```

Tooling

Beyond the compilation step, JSX does not require any special tools.

- Many editors already include reasonable support for JSX (Vim, Emacs js2-mode).
- Linting provides accurate line numbers after compiling without sourcemaps.
- Elements use standard scoping so linters can find usage of out-of-scope components.

Prior Work

JSX is similar to several other JavaScript embedded XML language proposals/projects. Some of the features of JSX that distinguish it from similar efforts include:

- JSX is a simple syntactic transform.
- JSX neither provides nor requires a runtime library.
- JSX does not alter or add to the semantics of JavaScript.

JSX is similar to HTML, but not exactly the same. See JSX gotchas for some key differences.

Post to Facebook Posting as Stoyan Stefanov (Change) **Comment**

Jakub Malinowski · Akademia Górniczo-Hutnicza

ternary if?: like they show here should be sufficient in most cases, you can even execute code this way.

Reply · 1 · Like · Follow Post · July 21 at 3:07am

Peter Riboprotein · Seattle, Washington

How should I make if statements with React?
For example, if I have `state.editing` I want to be able to provide one JSX block, and if it's not then I want to provide a second JSX block.

I could have an if statement outside of the JSX blocks, but then I end up repeating a lot of the parent objects. (My wrapping divs and such.)

Is there an existing pattern for this? I could make a custom block like this:
<if expression={this.state.foobar}><Children></if>
Is that recommend?

Reply · Like · Follow Post · July 19 at 6:19pm

Peter Riboprotein · Seattle, Washington

okay-- here's the pattern I'm using:

if expression
inner = <derp />.
else
inner = <herp />.

return `<div class="wrapping>.
{inner}.
`.

Reply · Like · Follow Post · Edited · July 20 at 1:06am

Christopher Chedeau · Follow · User Interface Engineer at Facebook

You can do

return
__{this.state.condition ? <first/> : <second />}

or