JSX High-Level

Prof. Andrew Sheehan

Boston University/MET
Computer Science Dept.

WHAT IS JSX

JavaScript XML

JSX KEY POINT

JSX is not HTML and it is not a String

PURPOSE OF JSX

Allows you to create React elements within your components.

You do not have to use it.

```
// Using JSX to express UI components
var dropdown =
  <Dropdown>
    A dropdown list
    <Menu>
      <MenuItem>Do Something</MenuItem>
      <MenuItem>Do Something Fun!</MenuItem>
      <MenuItem>Do Something Else</MenuItem>
    </Menu>
  </Dropdown>;

render(dropdown);
```

They are separate concepts

Note

JSX and React are two separate things. They're often used together, but you *can* use them independently of each other. JSX is a syntax extension, while React is a JavaScript library.

First Rule

To return multiple elements, you must wrap them with a single element.

```
<div>
  <h1>Hi There!</h1>
  <span>Welcome.</span>
</div>
```

Alternative Syntax

Instead of wrapping your elements with a div or …

You can use <> … </>

It is called an "Empty Fragment"

```
<>
    <h1>Welcome back!>
    <span>Your last login was December 1st, 2023 @ 7:00 PM EST</span>
</>
```

2nd Rule: Use Strict HTML Syntax

All tags that are empty – like image or link must be properly closed.

VARIABLE INTERPOLATION

```
const age = getClientAge();

const ageElem = <span>{age}</span>;

// expressions are possible, too
const nextAge = <span>{ getClientAge() +1 }</span>;
```

CONTROL STRUCTURES

JSX is allowed in control structures

```
function welcome(user = “guest”) {
    if (user === “guest”) {
        return <h1>Hello,{user}</h1>;
    } else {
        return <h1>Welcome back {user}!</h1>;
    }
}
// using welcome()…
render() {
    return ( {welcome()} );
}
```

Don't use quotes around curlies when embedding a Javascript expression in an attribute definition.

```
const element = <img src={user.avatarUrl}></img>;
```

NO QUOTES SURROUNDING ATTRIBUTES

WRITING COMMENTS

<!– This is a HTML comment -->

{ /* This is a JSX comment */ }

React uses camelCase with HTML attributes

`class` becomes `className` in JSX, and `tabindex` becomes `tabIndex`.

ATTRIBUTES: CAMELCASE

CHILD ELEMENTS

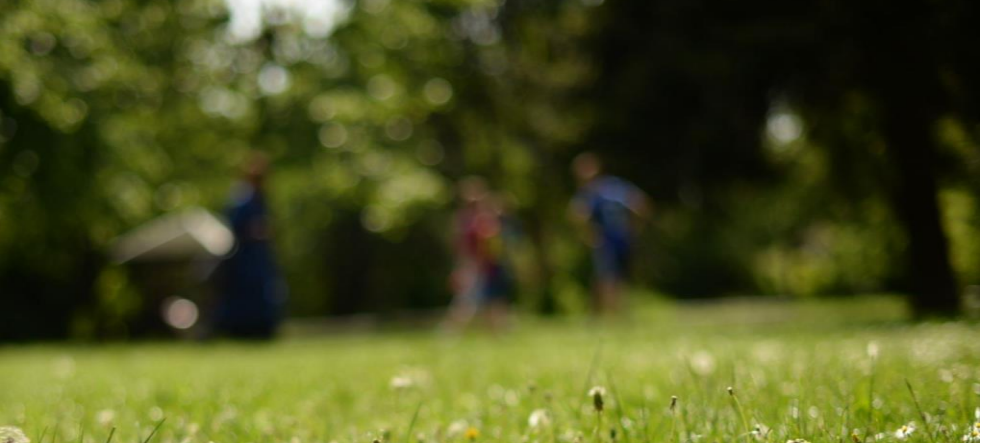
```
<section>
  <h1>News Today</h1>
  <article>
    <strong>I</strong>
      today's news....
  </article>
</section>
```

PREVENTING XSS

By default, React DOM escapes any values embedded in JSX before rendering

Everything converted to a string

Helps prevent XSS (cross-site-scripting).



Use a tool:

https://transform.tools/html-to-jsx

HTML

Settings

```
<!-- Hello world -->
<div class="awesome" style="border: 1px solid red">
  <label for="name">Enter your name: </label>
  <input type="text" id="name" />
</div>
<p>Enter your HTML here</p>
```

JSX

Copy

```
<>
  {/* Hello world */}
  <div className="awesome" style={{ border: "1px solid red"
    <label htmlFor="name">Enter your name: </label>
    <input type="text" id="name" />
  </div>
  <p>Enter your HTML here</p>
</>
```

CONVERTING HTML TO JSX

FUNCTIONS

JSX

Invoking functions within JSX: allowed.

```
const ele = (
 <h1>Good afternoon, {f1 (user)} </h1>
);
```

DYNAMIC VALUE INDICATION

```
<img
   src={imgSource}
   alt={imgAlt}
   className="bts blackpink"
   width="32"
/>
```

Objects and CSS can be used when you use double curlies in your JSX

```
<ul style={{
        backgroundColor: ‘hot pink',
        color: ‘grey'
}}>
    <li>BTS</li>
    <li>EXO</li>
    <li>MONSTA X</li>
    <li>Seventeen</li>
    <li>GOT7</li>
</ul>
```

DOUBLE CURLIES

{{ }}

Inline `style` properties are written in camelCase. For example, HTML `<ul style="background-color: black">` would be written as `<ul style={{ backgroundColor: 'black' }}>` in your component.

```
let person = {
    name: 'Andrew',
    course_count: 2
}

let content =
      <div>
         <span>{person.name}</span>
      </div>
```

DOUBLE CURLIES

{{ }}