skillcrush

THE BASICS OF WRITING REACT CODE

cheatsheet

INTRO

This little cheatsheet walks through the four aspects of writing React: JSX tags, comments, attributes, and elements! Keep this on your fridge, by your computer, or on your desktop to help you become a React master!

JSX TAGS

JSX tags work similarly to HTML tags! JSX tags help React build complex UIs. You can even add JavaScript code into your JSX tags, like this example:

```
1  const HeyGirl = (
2       <h1 className="large">Hey
3       girl! Today is {Date}. Have a great day!</h1>
4     );
```

JSX COMMENTS

You'll use **comments** to tell your colleagues or your future self what each piece of code is for. You can also use commenting to remove a snippet of code you don't need, or when you're debugging code to find which snippet isn't working properly.

To write a comment in JSX, you'll write {/* to start the comment and */} to end the comment.

EXAMPLE COMMENT FOR A COLLEAGUE:

```
const HeyGirl = (
     <h1 className="large">Hey
     girl!</h1>
    {/* Intro message for the homepage! */}
   );
```

EXAMPLE COMMENTED OUT CODE:

```
const HeyGirl = (
     <h1 className="large">Hey
     girl!</h1>
    {/* <h2 className="large">Bye,
     girl!</h2>*/}
   );
```

JSX ATTRIBUTES

Attributes define a property of an object or element, like a quantity or color. Attributes in JSX can be variable, meaning they can be updated based on an event, like a user clicking a checkbox.

In JSX, you write attributes by including the variable, like myVariable, then adding an equal sign, like in the examples below.

EXAMPLE STATIC ATTRIBUTE:

```
const elementWithNoChildren = <img
src="http://link-to-some-funny-cat-pic" />;
```

EXAMPLE VARIABLE ATTRIBUTE:

```
const divStyle = { backgroundColor: "blue" };

divStyle.backgroundColor = "red";
const whatColorWillIBe = (
  <div style={divStyle}>
    <h1>I'm a fancy title on a colorful background</h1>
  </div>
);
```

JSX EXPRESSIONS

Expressions are a line of code that evaluates to a single value, such as 6 + 7, which would evaluate to 13.

React lets you evaluate JSX expressions in the middle of a JSX tag, like a **<p>** or **<ul>** tag! Evaluating expressions in your JSX tags helps you build dynamic UIs that can update as variables change.

For instance, you might write an expression that pulls a user's first name and last name, then writes that in a JSX tag to deliver a customized greeting in the UI.

When writing expressions in JSX, you'll use **camelCase** notation. camelCase variable naming is a convention where you start each new word with a capital letter, except for the first word of a variable, like helloSkillCrusher.

Finally, when writing an expression in a JSX tag, be sure to put the expression between {curly braces}, like in the examples below!

EXAMPLE JSX EXPRESSIONS:

```
1  // this expression will render an h1 element with "Hey Skill
2  Crusher!" as the text
3
4  const renderVariables = <h1>Hey {firstName + " " +
5  lastName}!</h1>;
```

```
1  // this expression will render a p element with 8 as the
   text
2  const evaluateAnyExpressions = <p>{1 + 2 + 5}</p>;
```

```
1  function getTime() {
2    const now = new Date();
3    return now.getHours() + ":" + now.getMinutes() + ":" +
4  now.getSeconds();
   }
5  // this expression will render a span element with the
   current
6  time as the text
7  const renderTheResultOfAFunctionCall =
   <span>{getTime()}</span>;
```