Returning JSX

Element Tags

Values in JSX

Basic Element Props

Using Inline Styles

Returning JSX

There are two valid ways to return JSX from a function.

First, you can put the opening tag of the JSX block on the same line as the return. Use this style if you are returning a single line of JSX.

```
const App = () => {
  return <h1>Hi there!</h1>;
};
```

Second, you can use a set of parens. The opening paren must be on the same line as the return statement. Use this style if you are returning multiple lines of JSX.

```
const App = () => {
  return (
    <div>
      <h3>My Blog Post</h3>
      <p>Here is some content for my blog.</p>
    </div>
  );
};
```

Make sure you always put the opening JSX element or paren on the same line as the return statement. If you don't, Javascript will assume you want to return nothing at all!

Element Tags

When writing HTML, some elements are self closing, such as an input element. For example, with HTML you can write an input as `<input>`.

Self-closing tags in JSX will always have a `/` at the end of the opening tag.

```
const App = () => {
  return <input />;
};
```

You may only return one 'top level' JSX element from a component. A 'top level' element is the most parent element being returned from a component.

This component is returning a single top level element - the 'ul'. The top level element can have as many elements inside of it as needed.

```
const App = () => {
  return (
    <ul>
      <li>Red</li>
      <li>Green</li>
    </ul>
  );
};
```

If you need to return multiple elements, wrap them with a containing element, like a div.

```
const App = () => {
  return (
    <div>
      <h3>Colors</h3>
      <ul>
        <li>Red</li>
        <li>Green</li>
      </ul>
    </div>
  );
};
```

Sometimes, you don't want to insert an extra wrapping element, since it can conflict with some CSS styles. If you don't want to add an extra element, you can return wrap the content in a 'fragment', which looks like an empty JSX element.

```
const App = () => {
  // The empty brackets are a 'React Fragment'
  return (
    <>
      <h3>Colors</h3>
      <ul>
        <li>Red</li>
        <li>Green</li>
      </ul>
    </>
  );
};
```

Values in JSX

We can show Javascript variables in JSX by wrapping the variable with curly braces.

```
const App = () => {
  const count = 10;

  return <div>Value of count is {count}</div>;
};
```

JSX can display numbers and strings. The values `true`, `false`, `null`, and `undefined` are not displayed.

```
const App = () => {
  const count = 10;
  const name = 'Jane';
  const booleanTrue = true;
  const booleanFalse = false;
  const nullValue = null;
  const undefinedValue = undefined;

  return (
    <ul>
      <li>Number: {count}</li>
      <li>String: {name}</li>
      <li>True: {booleanTrue}</li>
      <li>False: {booleanFalse}</li>
      <li>Null: {nullValue}</li>
      <li>Undefined: {undefinedValue}</li>
    </ul>
  );
};
```

Curly braces can contain any valid Javascript expression, not just a reference to a variable.

```jsx
const App = () => {
  return <div>{1 + 1}</div>;
};
```

Basic Element Props

Attributes assigned to individual elements are referred to as 'props'.

We can assign just about any prop to a JSX element as we would an HTML element.

Props that expect a string should be written with double quotes.

```jsx
const App = () => {
  return (
    <div>
      <input type="date" />
      <div>
        <img src="https://picsum.photos/150/150" />
      </div>
    </div>
  );
};
```

We can also use curly braces when assigning props to refer to a Javascript variable! We *don't* wrap the curly braces with double quotes when doing this.

```jsx
const App = () => {
  const type = 'date';

  return <input type={type} />;
};
```

Some HTML attribute names are different when we write JSX. The most common is `class`. With JSX, we write `className` instead of `class`.

```jsx
const App = () => {
  return <button className="btn btn-primary">Submit</button>;
};
```

The JSX will still work if you write `class`, but React will display an error in your console.

If we need to assign several props to an element, it should be broken up into multiple lines.

```jsx
const App = () => {
  return (
    <img
      src="https://picsum.photos/150/150"
      alt="This is a randomly picked photo"
    />
  );
};
```

Using Inline Styles

We prefer to write CSS in React apps, but sometimes we just have to use inline styles, which are applied directly to an element.

In HTML, inline styles look like this:

```
<div style="height: 100px; width: 100px; background-color: lightblue;"></div>
```

With JSX, however, we provide styles in an object. If the style name has a dash in it, we remove the dash and capitalize the next letter.

```
const App = () => {
  return (
    <div
      style={{
        height: '100px',
        width: '100px',
        backgroundColor: 'lightblue',
      }}
    />
  );
};
```

Notice the two sets of curly braces. The outer set means that we are about to write some Javascript in JSX. The inner set creates the object.

Earlier, we saw that objects can't be displayed in JSX. The key word there is 'displayed'. We can't show an object on the screen, but we can provide an object as a prop to an element.

Any CSS rule can be set from a React component.

```
const App = () => {
  return (
    <p
      style={{
        fontWeight: 'bold',
        fontStyle: 'italic',
        color: 'blue',
      }}
    >
      Hi there
    </p>
  );
};
```