Props Review

Like HTML, React Components can be passed attributes, called "props"

The component gets them as arguments:

```
<MyComp name="Bao"/>
```

```
function MyComp(props) {
  return (<div>{props.name}</div>);
}
```

```
<div>Bao</div>
```

You can destructure like any object/function call:

```
function MyComp({ name }) {
  return (<div>{ name }</div>);
}
```

Children (tag contents)

To access JSX contents, use special prop "children":

```
<MyComp>This is some content</MyComp>
```

```
const MyComp = ({ children }) => {
  return (
    <div>I heard: <b>{children}</b></div>
  );
};
```

```
<div>I heard: <b>This is some content</b></div>
```

```
import Box from './Box';
import Cat from './Cat';
const Room = () => {
  return (
    <div className="room">
      <Box><Cat name="Maru"/></Box>
      <Box><Cat name="Grumpy"/></Box>
      <Box/>
    </div>
  );
};
export default Room;
```

```
const Box = ({ children }) => {
  const contents = children ? children : <div>Nothing</div>;
  return ( <div> A box contains: {contents} </div>);
};
export default Box;
```

```
const Cat = ({ name }) => (<div>{name}</div>);
export default Cat;
```

When to use children

- Some components are content
    - paragraphs, lists, forms
- Some components are modify other content
    - modal windows, collapsing sections

If you have a generic way to modify content

- either take content as a child
- or accept the content as a prop
    - same as using children, but not showing it!
- not all modifying UI is a "wrapper"
    - think of HTML implementation

Fragments

React Components return a single parent element OR are a fragment

What is a fragment?

When you don't want a container element

A component often maps to an HTML element and any descendants

But not always

You might want multiple elements with no container

- because your component is an abstraction inside an existing container
- because your component is something not made of a single container

Fragment as the Solution

A React Fragment solves that desire

- fragment is the container
- but doesn't map to an HTML element
- only needed as the containing element of a component

The Wordy way

```
function Demo() {
  return (
    <React.Fragment>
      <p>
        These p tags will have no containing
        element from this component
      </p>
      <p>And React will not complain</p>
    </React.Fragment>
  );
}
```

The concise way

Empty tags are automatically treated as a fragment

```
function Demo() {
  return (
    <>
      <p>
        These p tags will have no containing
        element from this component
      </p>
      <p>And React will not complain</p>
    </>
  );
}
```

Why ever be wordy?

The concise syntax is great

But if you ever need a `key` prop on the fragment you will need the wordier syntax.

Can also:

```
import { Fragment } from 'react';

function Demo() {
  return (
    <Fragment>
      <p>Whatever</p>
    </Fragment>
  );
}
```

Summary - Children prop

Any content within a component tag

- is passed to that component as the `children` prop
- will only render if that component outputs the `children` prop

Use this for components that are wrappers of other content

Summary - Fragment

Fragments allow a component to return content that

- is wrapped in a single container (the fragment)
- does not render a single container element in the HTML

Fragments are only the single wrapping container

- not used elsewhere

Use whenever the wrapping container

- has no semantic meaning
- is not a target for styling

Summary - Fragment syntax

Fragments:

- can use `<> </>` syntax
  - unless a `key` prop is needed
  - then use either `<React.Fragment> </React.Fragment>`
  - or `<Fragment> </Fragment>`
  - depending on what you want to import