**Understanding the Props**

Props are arguments passed into React components.

Props are passed to components via HTML attributes.

props stands for properties.

**React Props**

React Props are like function arguments in JavaScript and attributes in HTML.

To send props into a component, use the same syntax as HTML attributes:

Example

**Add a "brand" attribute to the Car element:**

const myelement = <Car brand="Ford" />;

The component receives the argument as a props object

Example

**Use the brand attribute in the component:**

function Car(props) {

return <h2>I am a { props.brand }!</h2>;

}

**Pass Data**

Props are also how you pass data from one component to another, as parameters.

**Example**

Send the "brand" property from the Garage component to the Car component:

function Car(props) {

return <h2>I am a { props.brand }!</h2>;

}

function Garage() {

return (

<>

<h1>Who lives in my garage?</h1>

<Car brand="Ford" />

</>

);

}

ReactDOM.render(<Garage />, document.getElementById('root'));

If you have a variable to send, and not a string as in the example above, you just put the variable name inside curly brackets:

**Example**

**Create a variable named carName and send it to the Car component:**

function Car(props) {

return <h2>I am a { props.brand }!</h2>;

}

function Garage() {

const carName = "Ford";

return (

<>

<h1>Who lives in my garage?</h1>

<Car brand={ carName } />

</>

);

}

ReactDOM.render(<Garage />, document.getElementById('root'));

**Or if it was an object:**

Example

**Create an object named carInfo and send it to the Car component:**

function Car(props) {

return <h2>I am a { props.brand.model }!</h2>;

}

function Garage() {

const carInfo = { name: "Ford", model: "Mustang" };

return (

<>

<h1>Who lives in my garage?</h1>

<Car brand={ carInfo } />

</>

);

}

ReactDOM.render(<Garage />, document.getElementById('root'));

**React State**

React components has a built-in state object.

The state object is where you store property values that belongs to the component.

When the state object changes, the component re-renders.

**Creating the state Object**

The state object is initialized in the constructor:

The state object can contain as many properties as you like:

**Example:**

**Specify all the properties your component need:**

class Car extends React.Component {

constructor(props) {

super(props);

this.state = {

brand: "Ford",

model: "Mustang",

color: "red",

year: 1964

};

}

render() {

return (

<div>

<h1>My Car</h1>

</div>

);

}

}

**Changing the state Object**

To change a value in the state object, use the this.setState() method.

When a value in the state object changes, the component will re-render, meaning that the output will change according to the new value(s).

Example:

**Add a button with an onClick event that will change the color property:**

class Car extends React.Component {

constructor(props) {

super(props);

this.state = {

brand: "Ford",

model: "Mustang",

color: "red",

year: 1964

};

}

changeColor = () => {

this.setState({color: "blue"});

}

render() {

return (

<div>

<h1>My {this.state.brand}</h1>

<p>

It is a {this.state.color}

{this.state.model}

from {this.state.year

</p>

<button

type="button"

onClick={this.changeColor}

>Change color</button>

</div>

);

}

}

**Handling events like click events**

**React Events**

Just like HTML DOM events, React can perform actions based on user events.

React has the same events as HTML: click, change, mouseover etc.

**Adding Events**

React events are written in camelCase syntax:

* onClick instead of onclick.

React event handlers are written inside curly braces:

* onClick={shoot} instead of onClick="shoot()".

**React:**

<button onClick={shoot}>Take the Shot!</button>

**HTML:**

<button onclick="shoot()">Take the Shot!</button>

**Example:**

Put the shoot function inside the Football component:

function Football() {

const shoot = () => {

alert("Great Shot!");

}

return (

<button onClick={shoot}>Take the shot!</button>

);

}

ReactDOM.render(<Football />, document.getElementById('root'));

**Passing Arguments**

To pass an argument to an event handler, use an arrow function.

Example:

**Send "Goal!" as a parameter to the shoot function, using arrow function:**

function Football() {

const shoot = (a) => {

alert(a);

}

return (

<button onClick={() => shoot("Goal!")}>Take the shot!</button>

);

}

ReactDOM.render(<Football />, document.getElementById('root'));

**React Event Object**

Event handlers have access to the React event that triggered the function.

**In our example the event is the "click" event.**

Example:

Arrow Function: Sending the event object manually:

function Football() {

const shoot = (a, b) => {

alert(b.type);

/\*

'b' represents the React event that triggered the function,

in this case the 'click' event

\*/

}

return (

<button onClick={(event) => shoot("Goal!", event)}>Take the shot!</button>

);

}

ReactDOM.render(<Football />, document.getElementById('root'));