React Components

- Components are like functions that return HTML elements.
- Components are independent and reusable bits of code.
- They serve the same purpose as JavaScript functions, but work in isolation and return HTML.
- Components come in two types,
  - **Class components an**
  - **Function components**

Class Components

- A class component must include the extends React.Component statement.
- This statement creates an inheritance to **React.Component**, and gives your component access to React.Component's functions.
- The component also requires a **render()** method, this method returns HTML.

```
class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }}
```

Function Components

A Function component also returns HTML, and behaves much the same way as a Class component, but Function components can be written using much less code, are easier to understand, and will be preferred in this tutorial.

```
function Car() {

  return <h2>Hi, I am a Car!</h2>;}
```

Rendering a function Components

```
const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Car />);
```

Components in Components

```
function Car() {
  return <h2>I am a Car!</h2>;
}
function Garage() {
  return (
    <>
      <h1>Who lives in my Garage?</h1>
      <Car /> //Component Car in Component Garage
    </>
  );
}
const root = ReactDOM.createRoot(document.getElementById('root'));
 root.render(<Garage />);
```

Components in Files

This is the new file, we named it "Car.js":

```
function Car() {

  return <h2>Hi, I am a Car!</h2>;

}
export default Car;
```

Now we import the "Car.js" file in the application, and we can use the `Car` component as if it was created here.

```
import React from 'react';

import ReactDOM from 'react-dom/client';

import Car from './Car.js';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Car />);
```

React Props

- Props are arguments passed into React components.
- Props are passed to components via HTML attributes.
- 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:

```
const myElement = <Car brand="Ford" />;

function Car(props) {

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

}
```

Pass Data using Props

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

```
function Car(props) {
  return <h2>I am a { props.brand }!</h2>;
}
function Garage() {
  return (
    <>
      <h1>Who lives in my garage?</h1>
      <Car brand="Ford" />
    </>
  );
}
const root = ReactDOM.createRoot(document.getElementById('root'));
 root.render(<Garage />);
```

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() {
  const carName = "Ford";
  return (
    <>
      <h1>Who lives in my garage?</h1>
      <Car brand={ carName } />
    </>
  );
}
const root = ReactDOM.createRoot(document.getElementById('root'));

 root.render(<Garage />);
```

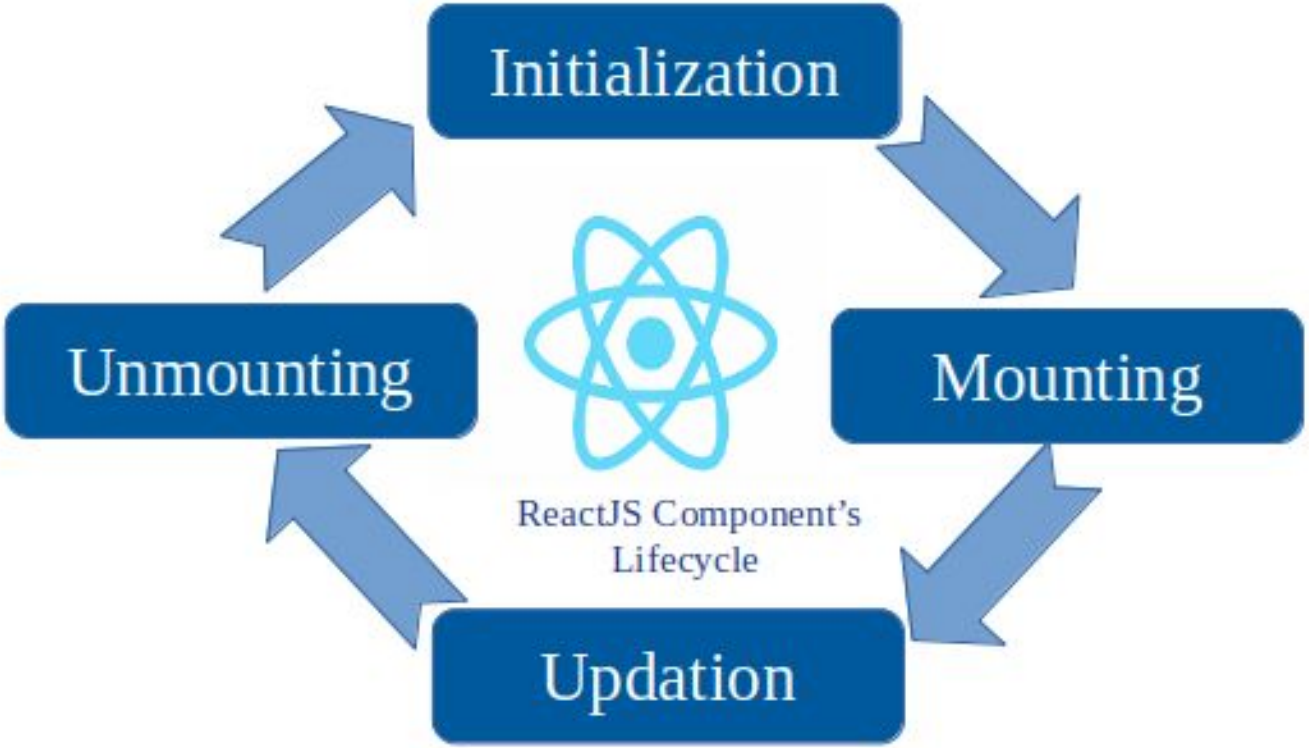
Create a variable named `carName` and send it to the `Car` component:

```jsx
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 } />
    </>
  );
}
const root = ReactDOM.createRoot(document.getElementById('root'));

 root.render(<Garage />);
```

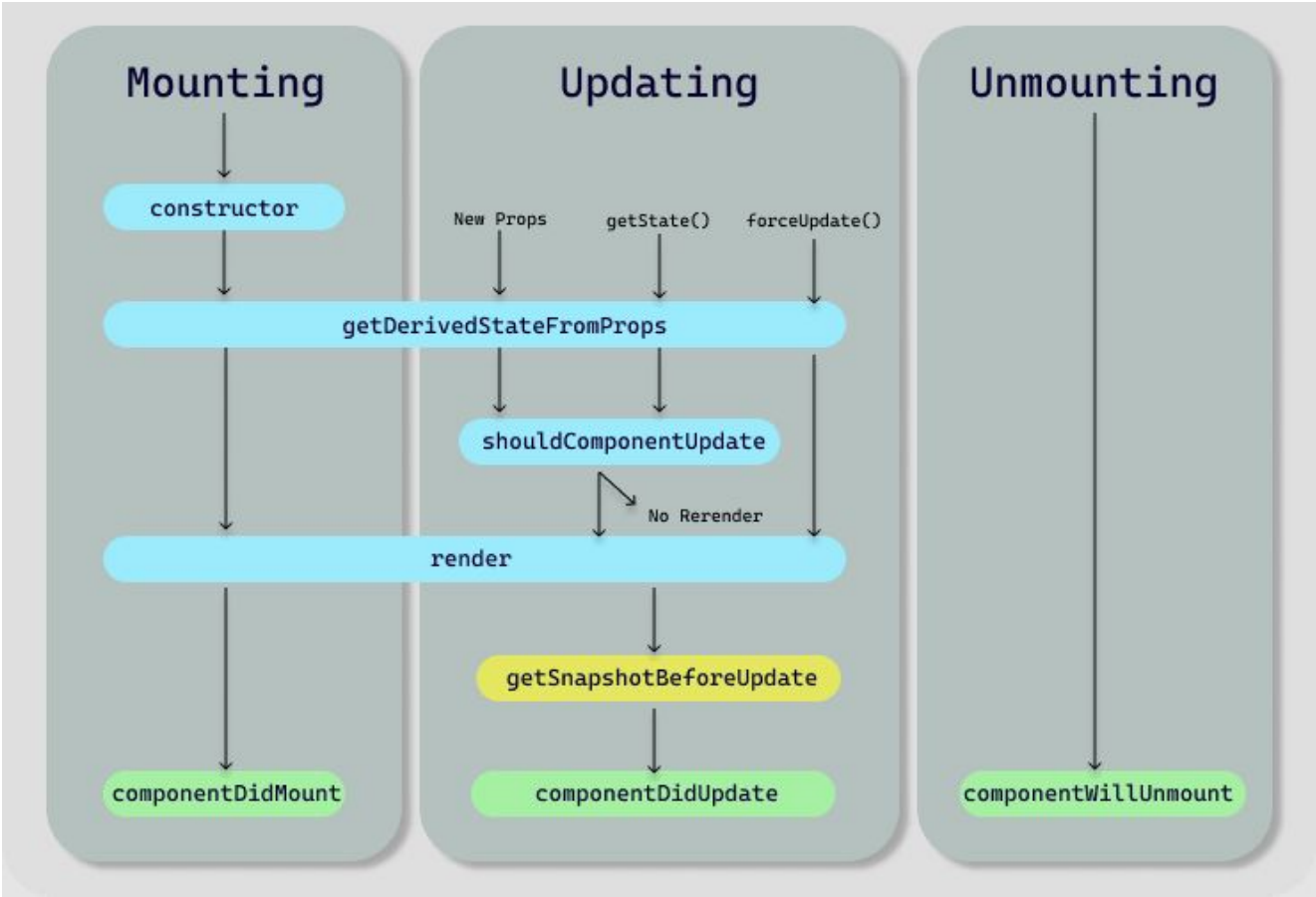
Create an object named `carInfo` and send it to the `Car` component:

React Component lifecycle



React Component lifecycle Methods

- **Mounting:** means putting elements into the DOM.
- **Updating:** The next phase in the lifecycle is when a component is *updated*. A component is updated whenever there is a change in the component's state or props.
- **Unmounting:**The next phase in the lifecycle is when a component is removed from the DOM, or *unmounting* as React likes to call it.



constructor()

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "yellow"};
  }
  render() {
    return (
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
    );
  }
}
 ReactDOM.render(<Header />, document.getElementById('root'));
```

getDerivedStateFromProps()

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "yellow"};
  }
  static getDerivedStateFromProps(props, state) {
    return {favoritecolor: props.favcol };
  }
  render() {
    return (
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
    );
  }
}
 ReactDOM.render(<Header favcol="yellow"/>, document.getElementById('root'));
```

render()

```
class Header extends React.Component {
  render() {
    return (
      <h1>This is the content of the Header component</h1>
    );
  }
}
ReactDOM.render(<Header />, document.getElementById('root'));
```

componentDidMount()

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  componentDidMount() {
    setTimeout(() => {
      this.setState({favoritecolor: "yellow"})
    }, 1000)
  }
  render() {
    return (
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
    );
  }
}
ReactDOM.render(<Header />, document.getElementById('root'));
```