Props

When React sees an element representing a user-defined component, it passes JSX attributes to this component as a single object. We call this object “props”.

```
function Student(props) {
  return ( <div>
        <h1>Hello, {props.name}</h1>;
        <h2>Your Roll: {props.roll}</h2>;
            </div> );
}
ReactDOM.render(  <Student name=“Rahul" roll=“101” />,  document.getElementById('root') );
ReactDOM.render(  <Student name={“Rahul“} roll=“101” />,  document.getElementById('root') );
ReactDOM.render(  <Student name=“Rahul" roll={100+1} />,  document.getElementById('root') );
```

JavaScript Expression as Props

If you pass no value for a prop, it defaults to true

Props

```
class Student extends React.Component {
  render() {
     return ( <div>
          <h1>Hello, {this.props.name}</h1>;
          <h2>Your Roll: {this.props.roll}</h2>;
                </div> );
       }
}
ReactDOM.render(  <Student name=“Rahul" roll=“101” />,  document.getElementById('root') );
```

Props

Whether you declare a component as a function or a class, it must never modify its own props. All React components must act like pure functions with respect to their props.

Pure Function

```
function sum(a, b) {
  return a + b;
}
```

Impure Function

```
function withdraw(account, amount) {
  account.total -= amount;
}
```

Props are Read-Only

Typechecking With PropTypes

npm install prop-types

To run typechecking on the props for a component, you can assign the special propTypes property.

Ex:-

```
import PropTypes from 'prop-types';
Student.propTypes = {
  name: PropTypes.string
};
```

Note –

- When an invalid value is provided for a prop, a warning will be shown in the JavaScript console.
- For performance reasons, propTypes is only checked in development mode.

Typechecking With PropTypes

PropTypes exports a range of validators that can be used to make sure the data you receive is valid.

```
optionalArray: PropTypes.array,
optionalBool: PropTypes.bool,
optionalFunc: PropTypes.func,
optionalNumber: PropTypes.number,
optionalObject: PropTypes.object,
optionalString: PropTypes.string,
optionalSymbol: PropTypes.symbol,
```

Required

```
import PropTypes from 'prop-types';
Student.propTypes = {
  name: PropTypes.string.isRequired
};
```

Default Prop Values

You can define default values for your props by assigning to the special defaultProps property.

```
Student.defaultProps = {
  name: ‘GeekyShows'
};
```

Children in JSX

In JSX expressions that contain both an opening tag and a closing tag, the content between those tags is passed as a special prop: props.children.

Ex:- <Student>I am child</Student>

```
props.children       // I am child
```