import PropTypes from 'prop-types';

MyComponent.propTypes = {

// You can declare that a prop is a specific JS primitive. By default, these

// are all optional.

optionalArray: PropTypes.array,

optionalBool: PropTypes.bool,

optionalFunc: PropTypes.func,

optionalNumber: PropTypes.number,

optionalObject: PropTypes.object,

optionalString: PropTypes.string,

optionalSymbol: PropTypes.symbol,

// Anything that can be rendered: numbers, strings, elements or an array

// (or fragment) containing these types.

optionalNode: PropTypes.node,

// A React element.

optionalElement: PropTypes.element,

// You can also declare that a prop is an instance of a class. This uses

// JS's instanceof operator.

optionalMessage: PropTypes.instanceOf(Message),

// You can ensure that your prop is limited to specific values by treating

// it as an enum.

optionalEnum: PropTypes.oneOf(['News', 'Photos']),

// An object that could be one of many types

optionalUnion: PropTypes.oneOfType([

PropTypes.string,

PropTypes.number,

PropTypes.instanceOf(Message)

]),

// An array of a certain type

optionalArrayOf: PropTypes.arrayOf(PropTypes.number),

// An object with property values of a certain type

optionalObjectOf: PropTypes.objectOf(PropTypes.number),

// An object taking on a particular shape

optionalObjectWithShape: PropTypes.shape({

color: PropTypes.string,

fontSize: PropTypes.number

}),

// You can chain any of the above with `isRequired` to make sure a warning

// is shown if the prop isn't provided.

requiredFunc: PropTypes.func.isRequired,

// A value of any data type

requiredAny: PropTypes.any.isRequired,

// You can also specify a custom validator. It should return an Error

// object if the validation fails. Don't `console.warn` or throw, as this

// won't work inside `oneOfType`.

customProp: function(props, propName, componentName) {

if (!/matchme/.test(props[propName])) {

return new Error(

'Invalid prop `' + propName + '` supplied to' +

' `' + componentName + '`. Validation failed.'

);

}

},

// You can also supply a custom validator to `arrayOf` and `objectOf`.

// It should return an Error object if the validation fails. The validator

// will be called for each key in the array or object. The first two

// arguments of the validator are the array or object itself, and the

// current item's key.

customArrayProp: PropTypes.arrayOf(function(propValue, key, componentName, location, propFullName) {

if (!/matchme/.test(propValue[key])) {

return new Error(

'Invalid prop `' + propFullName + '` supplied to' +

' `' + componentName + '`. Validation failed.'

);

}

})

};