HANDLING PROPS


YOU GET PROPS!
YOU GET PROPS!
EVERYBODY GET PROPS

Skicka ner via enskilda `props`

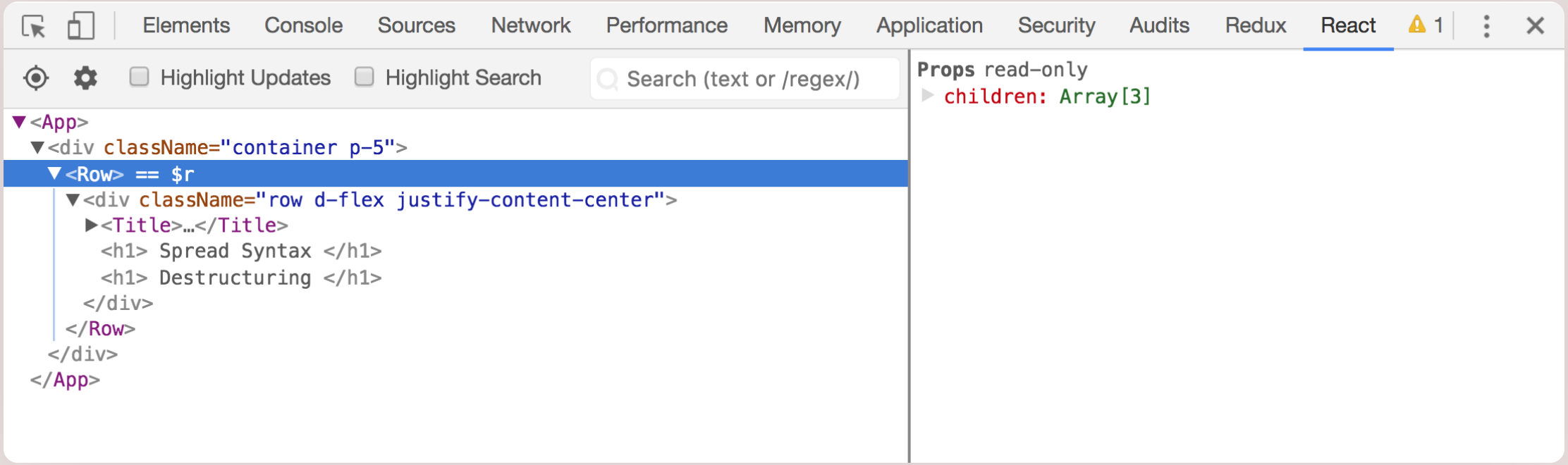
```
<Button
  type="primary"
  title="Login"
  isVisible={true} />
```

Vi kan skicka vidare objekt via `spread`

```
const obj = {
  id: 1249
  title: "The Title of This Thing"
}
<Card {...obj} /> //props.id, props.title
```

```
function Card(props){
  return <div key={props.id}> {props.title} </div>
}
```

Vi kan undersöka våra props i React DevTools



Samt vårt `state`

Det mesta ska skötas via `props` ändå

För att upptäcka fel med våra props så kan vi förstärka vår applikation med **Typechecking**

TYPECHECKING

JavaScript är ett dynamiskt typat språk

```
const a = 5;
let b = "What type?"
var c = true;
```

Vi har olika typer men JavaScript utvärdera detta vid runtime

Många andra språk är statiskt typade

```
int a = 5;
string b = "What type?";
bool c = true;
```

Det är vanligt att man vill förstärka vår applikation med att berätta för våra komponenter vilka värden som skickas ner.

Typechecking - kolla vilka typer vår data är

Detta gör vi med paketet `prop-types`

```
npm install --save prop-types
```

```
import PropTypes from 'prop-types';

function Text(props){
  return <h1> {props.text} </h1>
}

Text.propTypes = {
  text: PropTypes.string
}
```

```
Text.contextTypes = {
  optionalArray: PropTypes.array,
  optionalBool: PropTypes.bool,
  optionalFunc: PropTypes.func,
  optionalNumber: PropTypes.number,
  optionalObject: PropTypes.object,
  optionalString: PropTypes.string,
};
```

Allting är optional om vi inte lägger dit `.isRequired`

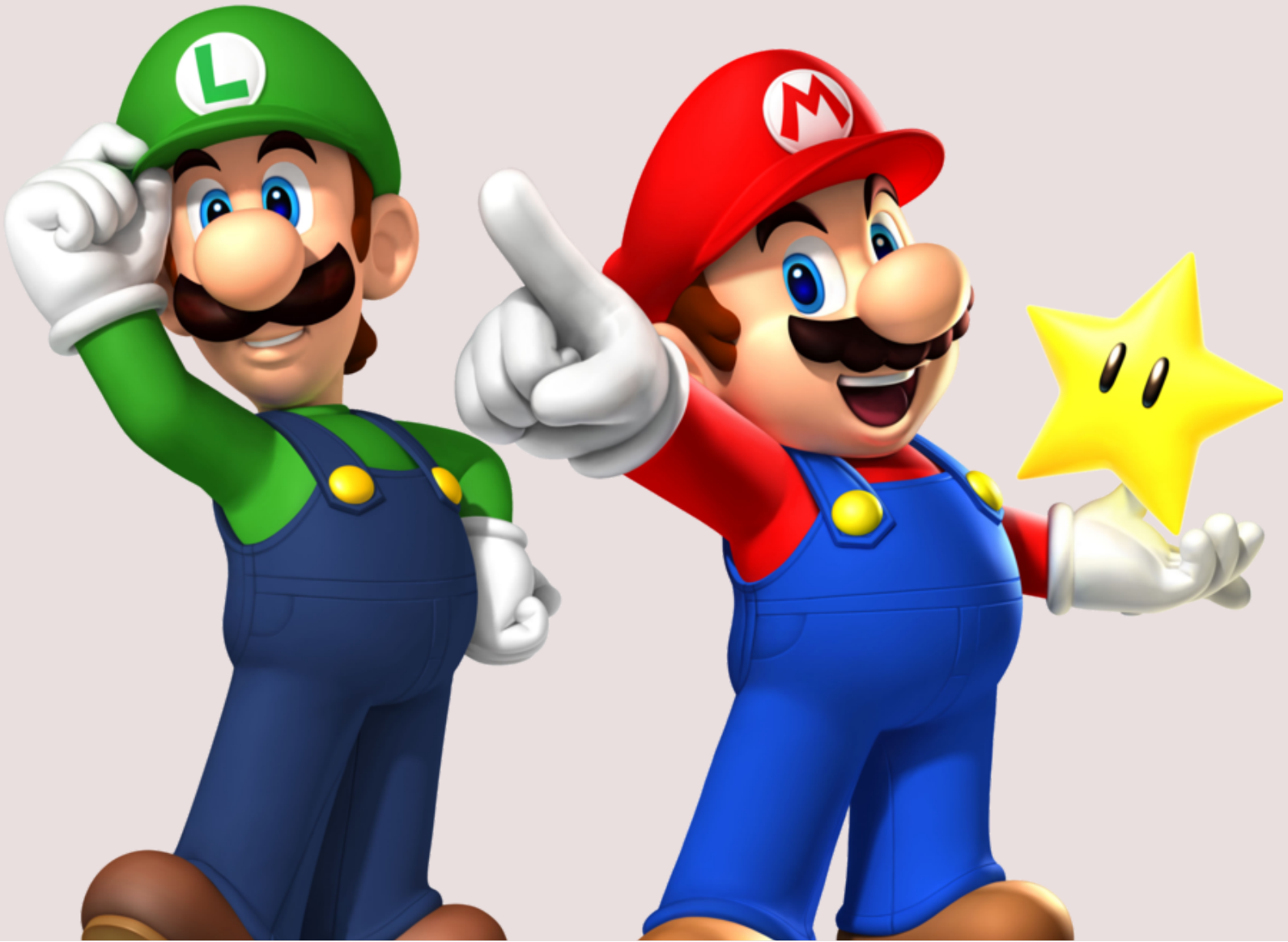
```
Text.contextTypes = {
  requiredArray: PropTypes.array.isRequired,
  requiredBool: PropTypes.bool.isRequired,
};
```

Använd `PROPTYPES`

Kan hjälpa dig undvika massa errors!

CONTEXT

`props` och `state`



`refs` och `context`

context är superglobal och kan användas för att skicka information långt ner i vår applikationsstruktur

Både `refs` och **`context`** ska användas så lite som möjligt.

`React Router`: standardbiblioteket för routing i React använder context

Definiera vilka värden som ska returneras

```
class App extends Component {
  //Inbyggd funktion i Component
  getChildContext() {
    return { contextString: "I'm in context!" };
  }
}
```

```
App.childContextTypes = {
  contextString: PropTypes.string //samt vilken typ
};
```

Kommer åt `context` via `this.context` i underkomponenter

```
class Text extends Component {
  render(){
    return <h1>{ this.context.contextString } </h1>
  }
}


Button.contextTypes = {
  contextString: PropTypes.string
};
```

Stateless

```
function Text(props, context){
  return <h1> { context.contextString } </h1>
}

Text.contextTypes = {
  contextString: PropTypes.string
};
```