STATE, PROPS & EVENTS

En komponent är en ES6-klass som returnerar JSX

JSX genererar HTML

```
class App extends Component{
  render(){
    return(
      <div>
        <h1>Hej!</h1>
      </div>
    );
  }
}
```

Vi behöver inte returnera JSX, vi kan returnera en komponent som returerar JSX

```
import Header from './Header';
class App extends Component{
  render(){
    return(
      <div>
        <Header />
      </div>
    );
  }
```

Modularitet! DEN ULTIMATA MAKTEN!

Vi hade också `props`, argument till dina komponenter

```
import Header from 'Header';
class App extends Component{
  render(){
    return(
      <div>
        <Header name="Jesper" />
      </div>
    );
  }
```

Alla props är tillgängliga i `this.props`

```
class Header extends Component{
  render(){
    return(
      <header>
        { /* "Jesper" */ }
        { this.props.name }
      </header>
    );
  }
}
```

`props` är immutable

Ska inte ändras av själva komponenter, enbart skickas nedåt

Hur ändrar vi props då? ... **state**

STATE

Varje class-komponent har både state och **props**

Props får inte ändras i en komponent

State får ändras hejvilt, det kommer att spåra ur.

Kom ihåg

Komponenter är bara funktioner

State och props är bara objekt

State och props är argument till dina komponenter som bestämmer hur ditt gränsnitt ser ut vid ett visst tillfälle.

Varje gång `state` ändras kallas `render()` på automatiskt

State skickas ner till underkomponenter och blir till props

VARJE GÅNG state och props ändras så uppdateras ditt gränssnitt

state kan skrivas som en class property

```
class App extends Component{
  state = {
    name: "Steffe"
  }
  render(){
    return <h1>{ this.state.name }</h1>;
  }
}
```

state ska enbart ändras genom `this.setState({})`

```
class App extends Component{
  state = {
    name: 'Jesper'
  }
  changeState = () => {
    this.setState({ name: 'Steffe' })
  }
  render(){
    return <h1>{ this.state.name }</h1>;
  }
}
```

KODEXEMPEL

STATELESS COMPONENTS

`ES6-classes` är bra för top-level components som ska hålla state och logik

De flesta komponenter är dock inte så komplicerade

Därför kan vi ersätta dem med "dumma komponenter"

```
function Header(props){
  return(
    <h1> {props.name} </h1>
  );
}
```

Observera att dessa komponenter inte har tillgång till `this`

Kan fortfarande vara interaktiv, få events ovanifrån

Kompakt ES6-variant

```
const Header = props => <h1> {props.name} </h1>
```

```
import React from 'react';
const Header = ({ name }) => <h1> { name } </h1>
export default Header;
```

Skapa innehåll dynamiskt

Vanliga for-loopar går!

Men oftast används:

- `map()`
- `reduce()`
- `filter()`

Om man inte har koll på `map` kan man skapa en array där alla element lagras i och rendera ut det elementet!

React keys

När vi skapar innehåll från arrayer förväntar sig React att kunna identifera varje element

React kräver att man sätter en `key`

```
let elements = array.map( (item, index ) => {
  return <li key={index} > { item } </li>;
});
```

EFTERMIDDAGEN

Övningar som finns under 'exercises' @ GitHub