

# React Best Practices

## 4 – Styling & CSS in JS

Timo Mämecke  
TH Köln // MI Master // Weaving the Web  
25. Juni 2019

# Inhalt

Oldschool CSS und React.

CSS Modules.

CSS-in-JS mit emotion.





# Oldschool CSS und React

# Standard CSS

```
import './styles.css'  
  
function PersonCard(props) {  
  return (  
    <div className="PersonCard">  
      <h1 className="PersonCard-name">{props.name}</h1>  
      <img src={props.avatar} />  
      <p className="PersonCard-motto">{props.motto}</p>  
    </div>  
  )  
}
```

Wir können "ganz normal" CSS verwenden.

# Konditionelle CSS-Klassen

```
import './styles.css'

function PersonCard(props) {
  let nameClass = 'PersonCard-name'
  if (props.position === 'boss') {
    nameClass += ' PersonCard-name--special'
  }

  return (
    <div className="PersonCard">
      <h1 className="PersonCard-name">{props.name}</h1>
      <img src={props.avatar} />
      <p className="PersonCard-motto">{props.motto}</p>
    </div>
  )
}
```



# Konditionelle CSS-Klassen mit "classnames"

```
import classNames from 'classnames'
import './styles.css'

function PersonCard(props) {
  const nameClass = classNames(
    'PersonCard-name',
    { 'PersonCard-name--special': props.position === 'boss' }
  )

  return (
    <div className="PersonCard">
      <h1 className={nameClass}>{props.name}</h1>
      <img src={props.avatar} />
      <p className="PersonCard-motto">{props.motto}</p>
    </div>
  )
}
```





# CSS Architekturen

Architekturen wie BEM und SuitCSS kapseln Styling-Regeln...

...durch Naming Conventions.

Brauchen wir noch CSS Architekturen?

# CSS Modules



# CSS Modules Beispiel

```
import styles from './styles.module.css'

function PersonCard(props) {
  return (
    <div className={styles.card}>
      <h1 className={styles.name}>{props.name}</h1>
      <img src={props.avatar} />
      <p className={styles.motto}>{props.motto}</p>
    </div>
  )
}
```

```
.card {
  padding: 20px;
  border: 1px solid black;
}

.name {
  font-size: 20px;
}

.motto {
  font-style: italic;
}
```

# CSS Modules

Bieten eine Mischung aus "oldschool CSS" und React Components.

Normale CSS Dateien nutzen.

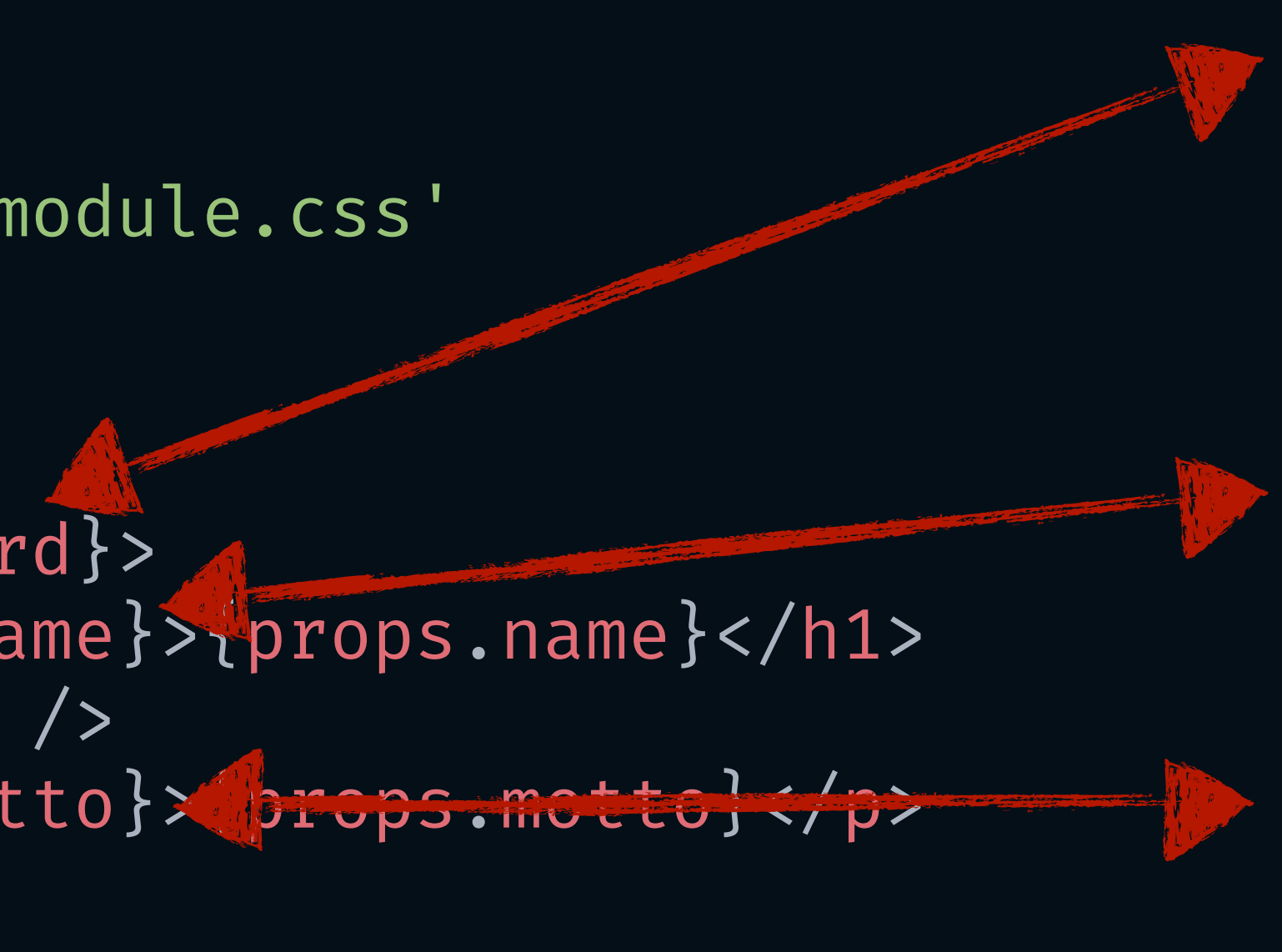
Mit dem Vorteil: Klassennamen sind nur innerhalb einer Komponente gültig.

\* Siehe: <https://github.com/css-modules/css-modules>  
Create React App unterstützt CSS Modules, wenn die Datei mit '.module.css' endet.

# CSS Modules Beispiel

```
import React from 'react'
import styles from './styles.module.css'

function PersonCard(props) {
  return (
    <div className={styles.card}>
      <h1 className={styles.name}>{props.name}</h1>
      <img src={props.avatar} />
      <p className={styles.motto}>{props.motto}</p>
    </div>
  )
}
```



The diagram consists of three red arrows pointing from the code on the left to the CSS styles on the right. The first arrow points from `styles.card` in the `<div>` tag to the `.card` selector. The second arrow points from `styles.name` in the `<h1>` tag to the `.name` selector. The third arrow points from `styles.motto` in the `<p>` tag to the `.motto` selector.

```
.card {
  padding: 20px;
  border: 1px solid black;
}

.name {
  font-size: 20px;
}

.motto {
  font-style: italic;
}
```



# CSS Modules Beispiel

## mit konditionellen Klassen

```
import React from 'react'
import classNames from 'classnames'

import styles from './styles.module.css'

function PersonCard(props) {
  const nameClass = classNames(
    styles.name,
    { [styles.special]: props.position === 'boss' }
  )

  return (
    <div className={styles.card}>
      <h1 className={nameClass}>{props.name}</h1>
      <img src={props.avatar} />
      <p className={styles.motto}>{props.motto}</p>
    </div>
  )
}
```



Ich bin kein Fan.

# CSS in JS



# Überlegungen hinter CSS in JS

Wieso nicht inline styles?

Warum nutzen wir nicht das "style" Attribut?

- Keine Pseudoelemente.
- Keine Hover-States.
- Performance Probleme.

Dann halt inline styles, aber ohne das "style" Attribut!

# Beliebte CSS in JS Libraries

styled-components<sup>1</sup>

- konzentriert sich hauptsächlich auf das Konzept der "styled components"

emotion<sup>2</sup>

- versucht alle styling-Methoden abzubilden

Beide generieren dynamisch CSS Klassen

<sup>1</sup> <https://www.styled-components.com>

<sup>2</sup> <https://emotion.sh/>

# emotion

```
import { css } from 'emotion'

function PersonCard(props) {
  return (
    <div className={css({ padding: 20, border: '1px solid black' })}>
      <h1 className={css({
        fontSize: props.position === 'boss' ? 30 : 20
      })}>
        {props.name}
      </h1>
      <img src={props.avatar} />
      <p className={css({ fontStyle: 'italic' })}>{props.motto}</p>
    </div>
  )
}
```



# emotion

```
import { css } from 'emotion'

function PersonCard(props) {
  return (
    <div className={css({ padding: 20, border: '1px solid black' })}>
      <h1 className={css({
        fontSize: props.position === 'boss' ? 30 : 20
      })}>
        {props.name}
      </h1>
      <img src={props.avatar} alt="Avatar" data-bbox="468 604 531 711" />
      <p className={css({ fontStyle: 'italic' })}>{props.motto}</p>
    </div>
  )
}
```

Kein unnötiges Naming mehr!

Dynamische Klassen sind viel besser!

# noch besser: babel-plugin-emotion

```
function PersonCard(props) {
  return (
    <div css={{ padding: 20, border: '1px solid black' }}>
      <h1 css={{
        fontSize: props.position === 'boss' ? 30 : 20
      }}>
        {props.name}
      </h1>
      <img src={props.avatar} />
      <p css={{ fontStyle: 'italic' }}>{props.motto}</p>
    </div>
  )
}
```

babel-plugin-emotion transformiert automatisch "css" prop in "className".

# emotion und styled components

```
import styled from '@emotion/styled'

const Card = styled.div`
  padding: 20px;
  border: 1px solid black;
`

const Name = styled.h1`
  font-size:
    ${props => props.special ? '30px' : '20px'};
`

const Motto = styled.p`
  font-style: italic;
`

function PersonCard(props) {
  return (
    <Card>
      <Name
        special={props.position === 'boss'}
      >
        {props.name}
      </Name>
      <img src={props.avatar} />
      <Motto>{props.motto}</Motto>
    </Card>
  )
}
```

"styled components" sind kleine React Components, die nur aus styling bestehen.

# Meine Empfehlung

Emotion ist ein guter allrounder.

css prop mit babel-plugin-emotion ist super, wenn man babel-plugins kontrollieren kann.<sup>1</sup>

Markup aufräumen mit styled components, wenn nötig.

<sup>1</sup> Wenn nicht, kann man das jsx-pragma nutzen: <https://emotion.sh/docs/css-prop#jsx-pragma>



# tl;dr

Es ist **möglich**, alte CSS-Methodiken zu verwenden.

CSS Modules klingen cool, werden aber nervig.

Beste Developer Experience mit CSS in JS (z.B. mit emotion).