Styled Components

6기_이희제

Installation

npm install --save styled-components

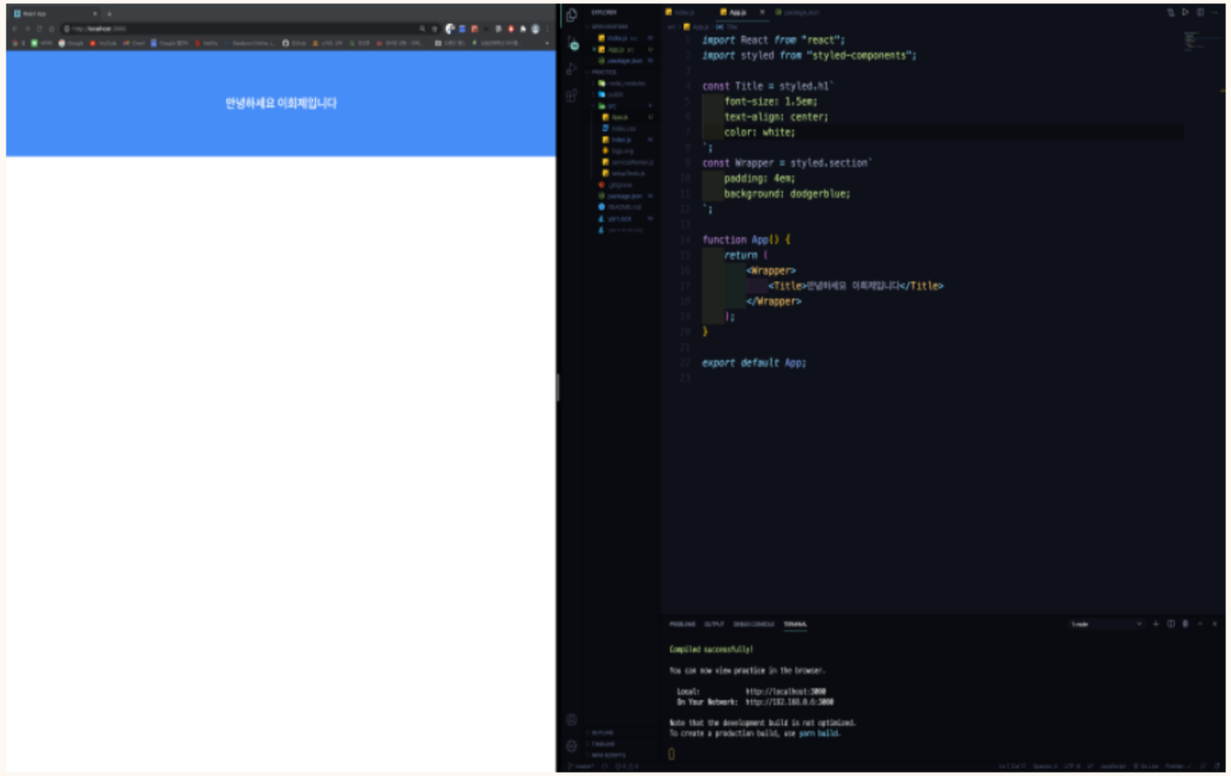
yarn add styled-components

Example

```
// Create a Title component that'll render an <h1> tag with some styles
const Title = styled.h1`
  font-size: 1.5em;
  text-align: center;
  color: palevioletred;
`;

// Create a Wrapper component that'll render a <section> tag with some styles
const Wrapper = styled.section`
  padding: 4em;
  background: papayawhip;
`;

// Use Title and Wrapper like any other React component - except they're styled!
render(
  <Wrapper>
    <Title>
      Hello World!
    </Title>
  </Wrapper>
);
```



Adapting based on props

```
const Button = styled.button`
  /* Adapt the colors based on primary prop */
  background: ${props => props.primary ? "palevioletred" :
"white"};
  color: ${props => props.primary ? "white" : "palevioletred"};

  font-size: 1em;
  margin: 1em;
  padding: 0.25em 1em;
  border: 2px solid palevioletred;
  border-radius: 3px;
`;

render(
  <div>
    <Button>Normal</Button>
    <Button primary>Primary</Button>
  </div>
);
```

props 값을 primary 로 주면 배경과 글자색이 바뀐다.

Extending Styles

```
// The Button from the last section without the interpolations
const Button = styled.button`
  color: palevioletred;
  font-size: 1em;
  margin: 1em;
  padding: 0.25em 1em;
  border: 2px solid palevioletred;
  border-radius: 3px;
`;

// A new component based on Button, but with some override
styles
const TomatoButton = styled(Button)`
  color: tomato;
  border-color: tomato;
`;

render(
  <div>
    <Button>Normal Button</Button>
    <TomatoButton>Tomato Button</TomatoButton>
  </div>
);
```

기본적인 버튼 스타일 컴포넌트를 생성

Tomato Button 은 Button을 기반으로 생성됨.

스타일 요소를 덮어쓰거나 새로 쓸 수 있음.

'as'

```
const Button = styled.button`
  display: inline-block;
  color: palevioletred;
  font-size: 1em;
  margin: 1em;
  padding: 0.25em 1em;
  border: 2px solid palevioletred;
  border-radius: 3px;
  display: block;
`;

const TomatoButton = styled(Button)`
  color: tomato;
  border-color: tomato;
`;

render(
  <div>
    <Button>Normal Button</Button>
    <Button as="a" href="/">Link with Button styles</Button>
    <TomatoButton as="a" href="/">Link with Tomato Button
styles</TomatoButton>
  </div>
);
```


```
const Button = styled.button`
  display: inline-block;
  color: palevioletred;
  font-size: 1em;
  margin: 1em;
  padding: 0.25em 1em;
  border: 2px solid palevioletred;
  border-radius: 3px;
  display: block;
`;

const ReversedButton = props => <Button {...props} children=
{props.children.split('').reverse()} />

render(
  <div>
    <Button>Normal Button</Button>
    <Button as={ReversedButton}>Custom Button with Normal Button
styles</Button>
  </div>
);
```

Styling Component

```
// This could be react-router-dom's Link for example
const Link = ({ className, children }) => (
  <a className={className}>
    {children}
  </a>
);

const StyledLink = styled(Link)`
  color: palevioletred;
  font-weight: bold;
`;

render(
  <div>
    <Link>Unstyled, boring Link</Link>
    <br />
    <StyledLink>Styled, exciting Link</StyledLink>
  </div>
);
```

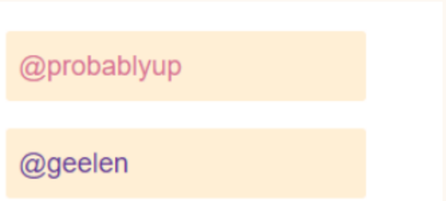
Unstyled, boring Link
Styled, exciting Link

className prop를 넘겨주면서 component 를 꾸밀 수 있음

Passed props

```
// Create an Input component that'll render an <input> tag with
some styles
const Input = styled.input`
  padding: 0.5em;
  margin: 0.5em;
  color: ${props => props.inputColor || "palevioletred"};
  background: papayawhip;
  border: none;
  border-radius: 3px;
`;

// Render a styled text input with the standard input color, and
one with a custom input color
render(
  <div>
    <Input defaultValue="@probablyup" type="text" />
    <Input defaultValue="@geelen" type="text"
inputColor="rebeccapurple" />
  </div>
);
```



props 인 inputColor를 넘겨준다.

Coming from CSS

기존 방식(css module)

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

export default class Counter extends React.Component {
  state = { count: 0 }

  increment = () => this.setState({ count: this.state.count + 1 })
  decrement = () => this.setState({ count: this.state.count - 1 })

  render() {
    return (
      <div className={styles.counter}>
        <p className={styles.paragraph}>{this.state.count}</p>
        <button className={styles.button} onClick={this.increment}>
          +
        </button>
        <button className={styles.button} onClick={this.decrement}>
          -
        </button>
      </div>
    )
  }
}
```

Coming from CSS

Styled-component 방식

```
import React from 'react'
import styled from 'styled-components'

const StyledCounter = styled.div`
  /* ... */
`
const Paragraph = styled.p`
  /* ... */
`
const Button = styled.button`
  /* ... */
`

export default class Counter extends React.Component {
  state = { count: 0 }

  increment = () => this.setState({ count: this.state.count + 1 })
  decrement = () => this.setState({ count: this.state.count - 1 })

  render() {
    return (
      <StyledCounter>
        <Paragraph>{this.state.count}</Paragraph>
        <Button onClick={this.increment}>+</Button>
        <Button onClick={this.decrement}>-</Button>
      </StyledCounter>
    )
  }
}
```

Recommended way:

```
const StyledWrapper = styled.div`
  /* ... */
`

const Wrapper = ({ message }) => {
  return <StyledWrapper>{message}</StyledWrapper>
}
```

Bad way:

```
const Wrapper = ({ message }) => {
  // WARNING: THIS IS VERY VERY BAD AND SLOW, DO NOT DO THIS!!!
  const StyledWrapper = styled.div`
    /* ... */
  `

  return <StyledWrapper>{message}</StyledWrapper>
}
```

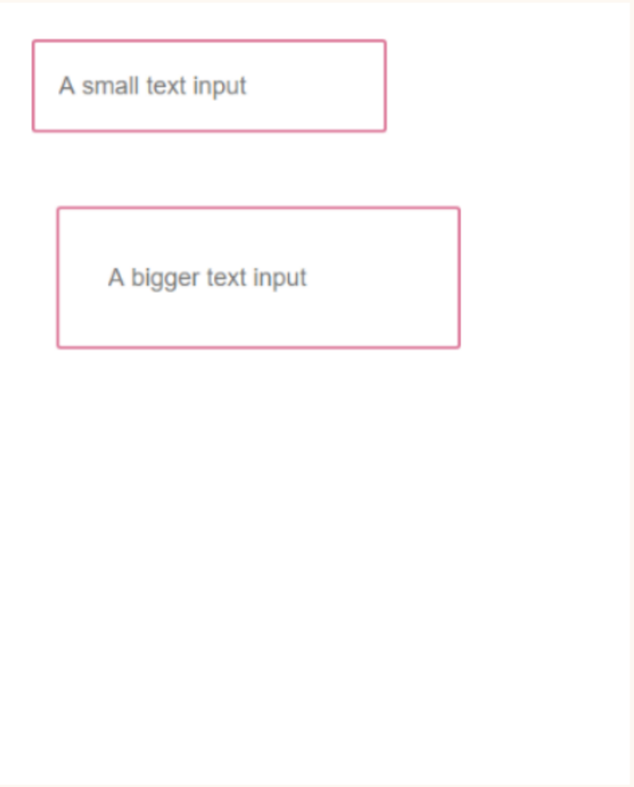
.attrs

```
const Input = styled.input.attrs(props => ({
  // we can define static props
  type: "text",

  // or we can define dynamic ones
  size: props.size || "1em",
}))`
  color: palevioletred;
  font-size: 1em;
  border: 2px solid palevioletred;
  border-radius: 3px;

  /* here we use the dynamically computed prop */
  margin: ${props => props.size};
  padding: ${props => props.size};
`;

render(
  <div>
    <Input placeholder="A small text input" />
    <br />
    <Input placeholder="A bigger text input" size="2em" />
  </div>
);
```



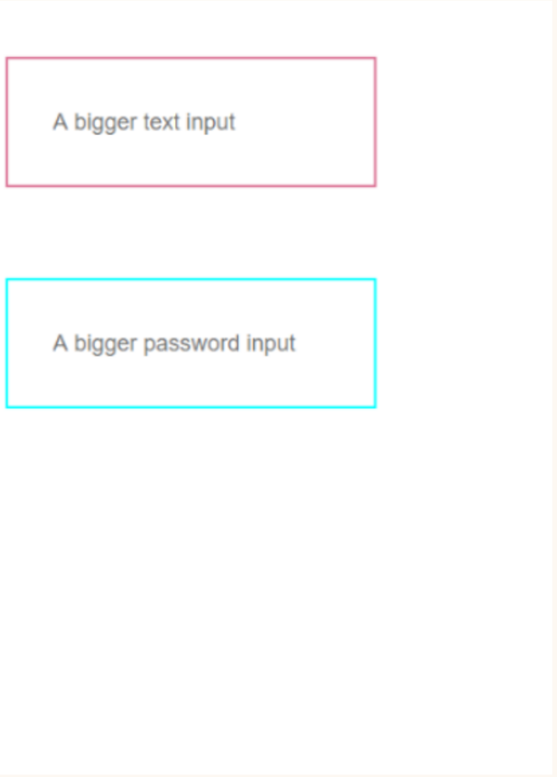
className prop를 넘겨주면서 component 를 꾸밀 수 있음

overriding .attrs

```
const Input = styled.input.attrs(props => ({
  type: "text",
  size: props.size || "1em",
}))`
  border: 2px solid palevioletred;
  margin: ${props => props.size};
  padding: ${props => props.size};
`;

// Input's attrs will be applied first, and then this attrs obj
const PasswordInput = styled(Input).attrs({
  type: "password",
})`
  // similarly, border will override Input's border
  border: 2px solid aqua;
`;

render(
  <div>
    <Input placeholder="A bigger text input" size="2em" />
    <br />
    {/* Notice we can still use the size attr from Input */}
    <PasswordInput placeholder="A bigger password input"
size="2em" />
  </div>
);
```



attrs 또한 가지고 와서 변경하고 추가할 수 있다.

Animations

```
// Create the keyframes
const rotate = keyframes`
  from {
    transform: rotate(0deg);
  }

  to {
    transform: rotate(360deg);
  }
`;

// Here we create a component that will rotate everything we
pass in over two seconds
const Rotate = styled.div`
  display: inline-block;
  animation: ${rotate} 2s linear infinite;
  padding: 2rem 1rem;
  font-size: 1.2rem;
`;

render(
  <Rotate>&lt; 🤣 &gt;</Rotate>
);
```

attrs 또한 가지고 와서 변경하고 추가할 수 있다.

감사합니다