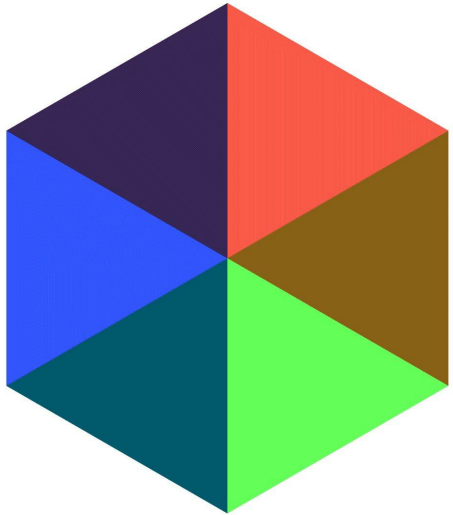
Start your app the better way with

Styled System

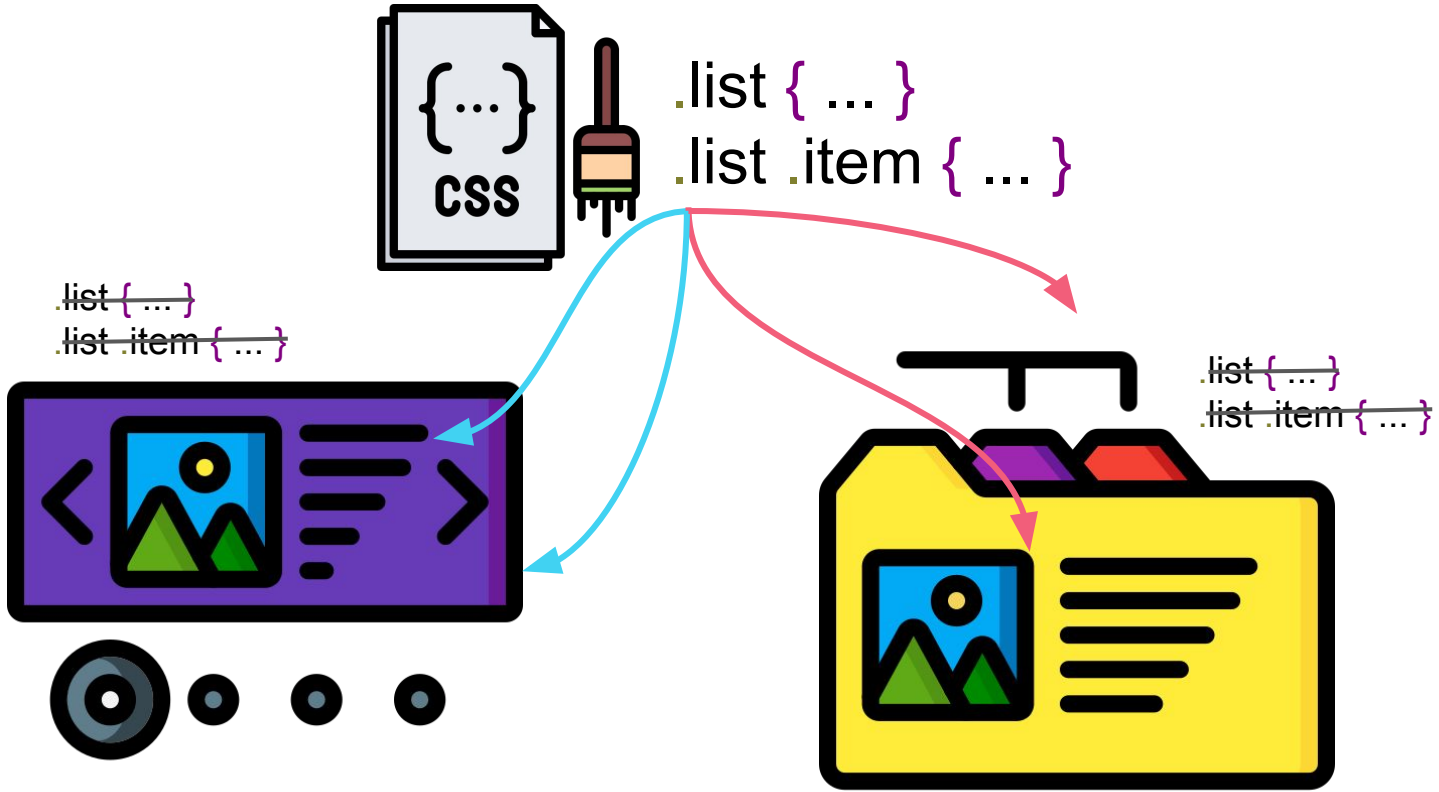


Agenda

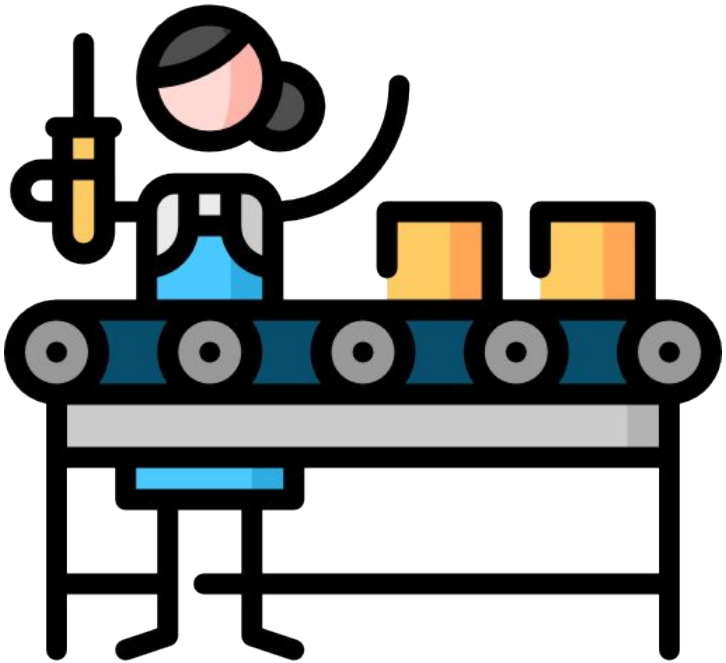
- History of CSS and it’s original sin
- The key problems of customizing UI component libraries’ styles
- Styled System
- QnA
- References

Everything in CSS is global.

Name collisions



Reusability



OOCSS

BEM

SMACSS

Block

A block component

.card-list

Element

A component of a block

.card-list__item

Modifier

State for a block or element.

.card-list__item--highlight

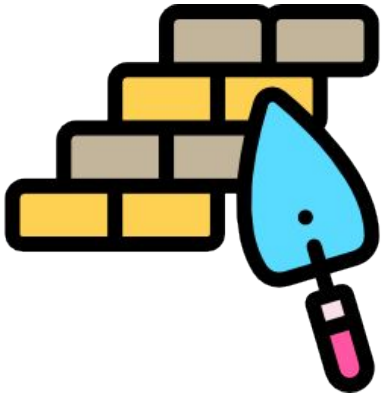
Preprocessor / Postprocessor

LESS
SASS
SCSS
PostCSS

```
.search-box-mixin(@scope) {
  .@{scope}-search-box {
    .search-input {
      border: 1px solid @gray;
    }
    .tooltip {
      margin: 8px 0 0 1px;
    }
  }
}
```

```
.search-box-mixin('page');
```

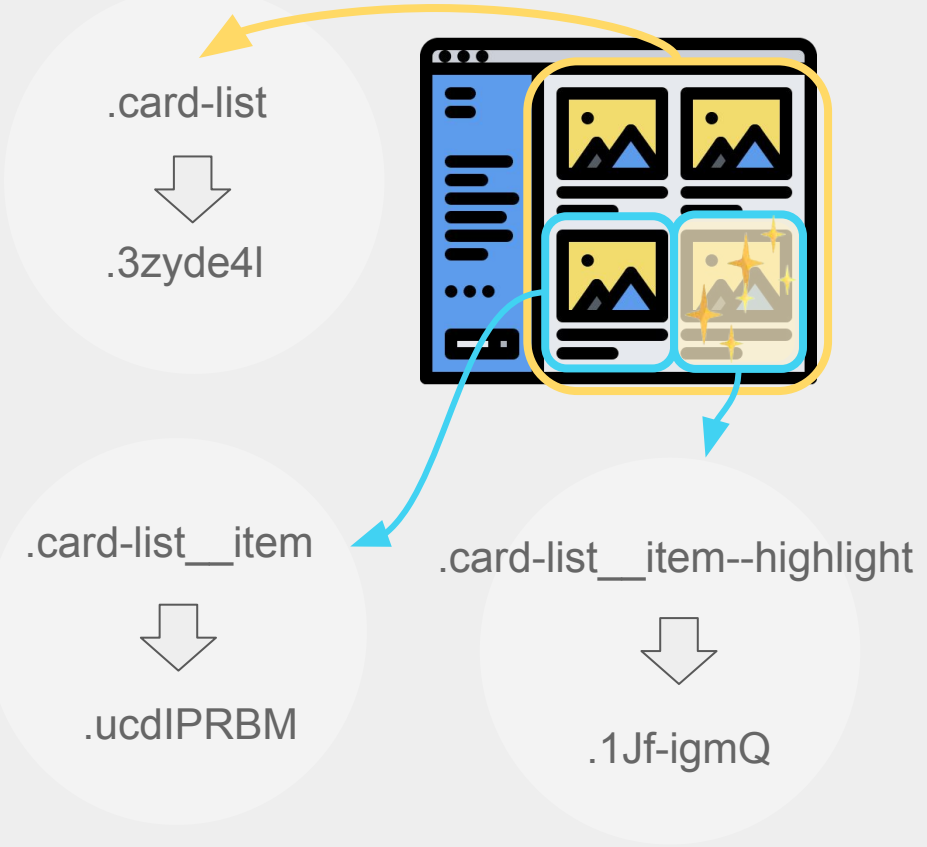
```
.page-search-box .search-input {
  border: 1px solid #ddd;
}

.page-search-box .tooltip {
  margin: 8px 0 0 1px;
}
```

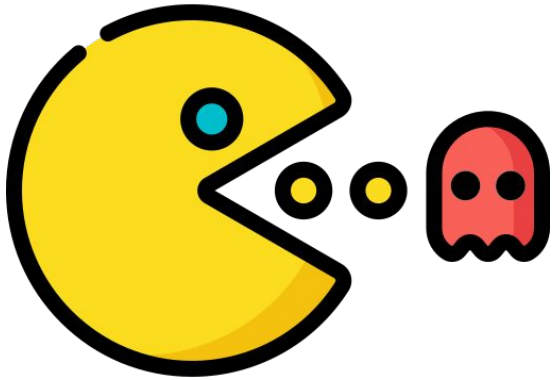


Components are the building blocks of modern web applications.

CSS Modules

Scope CSS rules locally by using tools to transform the class name with a hash.





CSS in JS

Write CSS in components.

```
import styled from
'styled-components';

const Button = styled.button`
  margin: 0 10px;
  padding: 10px;
  background: #fefefe;
  border-radius: 3px;
  border: 1px solid #ccc;
  color: #525252;
`
```

Still have some problems...

- Components with **inconsistent props**.
- How to efficiently implement **multiple themes**? Like SMASCC…
- How to efficiently implement **responsive** styles for different devices?
- How to **reduce redundant codes** for setting media queries or mapping props to css properties?

Recap

As an user, I care about...

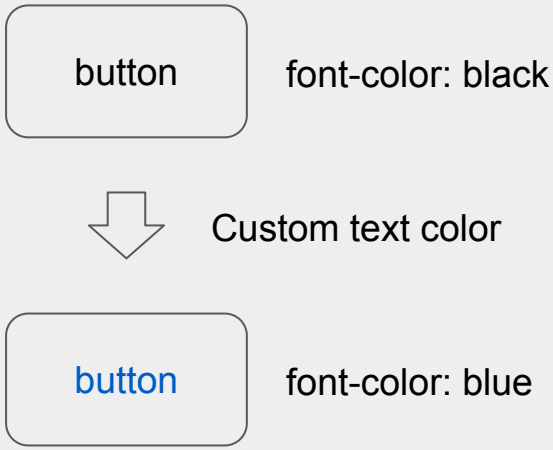
- Is it easy to custom styles?
- Does it support global theme styles and it is easy to custom?

As an developer, I care about...

- Develop UI with HIE’s design.

Ant Design

Step 1: Find the target element by using inspector.
Step 2: Extend LESS files to custom button styles.




```
@import '~antd/lib/button/style/index.less';
@btn-default-color: blue;
```

Ant Design

Step 1: Find the target element by using inspector.
Step 2: Wrap AntdButton by using CustomButton.

```
import styled from 'styled-components';
import { Button as AntdButton } from 'antd';

const CustomButton = styled(AntdButton)`
  span {
    color: blue;
  }
}`;

export default CustomButton;
```

Could it be easier?

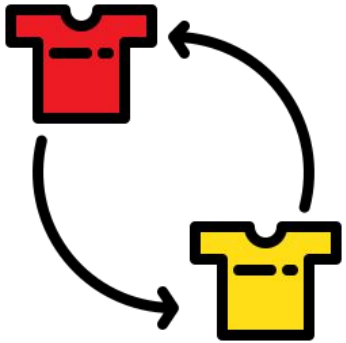
Start your app the better way with

Styled System

Why Styled System is good?

More concise codes

Consistency



Easy to custom styles

Mobile-First

More Concise Codes

Utility functions make codes short and clear,
expressing styles without unnecessary words

Utility functions (1/2)


```
<Box color='#000 bg='tomato' />
```

```
const Box = styled.div`
  margin: 15px 0;
  padding: 15px;
  color: ${(props) => props.color};
  background: ${(props) => props.bg};
  border-radius: 10px;
`;
```

```
const getStyles = ({ color, bg }) => ({
  color,
  background: bg,
});

const Box = styled.div`
  ${getColor};
  margin: 15px 0;
  padding: 15px;
  border-radius: 10px;
`;
```

Utility functions (2/2)

```
const getStyles = ({ color, bg }) => ({
  color,
  background: bg,
});

const Box = styled.div`
  ${getColor};
  margin: 15px 0;
  padding: 15px;
  border-radius: 10px;
`;
```

Styled System


```
import { color } from 'styled-system';

const Box = styled.div`
  ${color}
  margin: 15px 0;
  padding: 15px;
  border-radius: 10px;
`;
```

Consistency

Make components with consistent prop names

Inconsistent props

```
<Button color='black'>Click</Button>
```

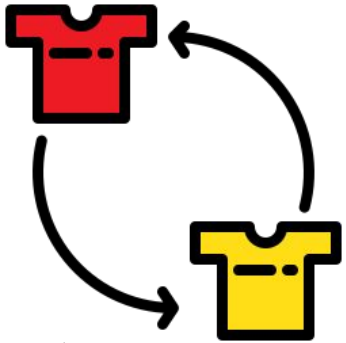
Use color utility function in Styled System

```
<Label fontColor='white'>$</Label>
```

```
<Button color='black'>Click</Button>
```

```
<Label color='white'>$</Label>
```

Easy to custom styles

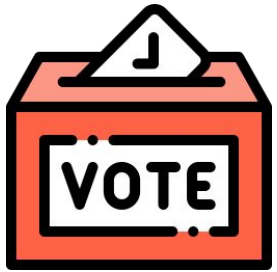


- Define global theme by a config object
- Define individual component styles by variants

Theming

Utilizing `<ThemeProvider>` and pass the theme object from root node to provide global theme.

```
<ThemeProvider theme={theme}>
  <Box color='black' bg='tomato'/>
</ThemeProvider>
```



```
const theme = {
  color: {
    black: '#333',
  },
  bg: {
    tomato: 'tomato',
  },
};
```

Define component styles in theme object

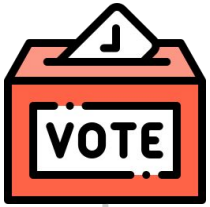
```
<Button variant="danger" size="large" />
```

```
const buttonStyle = variant({ key:
'buttons'});
const buttonSizeStyle = variant({
prop: 'size', key: 'buttons.size' });

const Button = styled.div`
  ${buttonStyle}
  ${buttonSizeStyle}
  padding: 15px;
`;
```

```
const theme =  {
  buttons: {
    danger: {
      color: 'white',
      background: '#f25e7a'
    },
    size: {
      default: { height: 50 },
      large: { height: 100 }
    }
  }
};
```

Variants

Use variants to define component styles.




```
<Box variant='primary' />
```


```
<Box variant='secondary'/>
```

```
import { variant } from 'styled-system';

const Box = styled('div')(
  variant({
    variants: {
      primary: {
        color: 'black',
        bg: 'tomato'
      },
      secondary: {
        color: 'black',
        bg: 'yellow'
      },
    },
  }),
);
```

Mobile-First

Create mobile-first responsive layouts with **array syntax**

Responsive styles

Create mobile-first responsive layouts with ease by using an array syntax.

```
.thing {
  font-size: 16px;
  width: 100%;
}

@media screen and (min-width: 40em) {
  font-size: 20px;
  width: 50%;
}

@media screen and (min-width: 52em) {
  font-size: 24px;
}
```

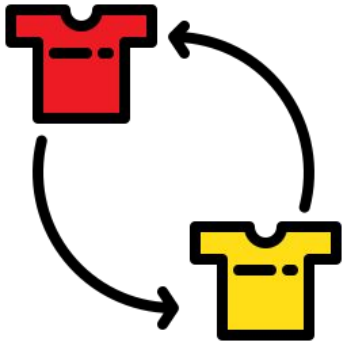
Styled System


```
<Thing
  fontSize={[ 16, 20, 24 ]}
  width={[1, 1/2]}
/>
```

Styled System is good!

More concise codes

Consistency



Easy to custom styles

Mobile-First

以前的我：

現在的我：

chakra

Build accessible React apps & websites with speed

Chakra UI is a simple, modular and accessible component library that gives you all the building blocks you need to build your React applications.

Get Started GitHub

Accessible

Chakra UI strictly follows WAI-ARIA standards. All components come with proper attributes and keyboard interactions out of the box.

Themeable

Quickly and easily reference values from your theme throughout your entire application, on any component.

Composable

Components were built with composition in mind. You can leverage any component to create new things.

Custom component styles


```
<Button>Button</Button>
```

```
<Button
  variantColor='green'
  color='#ddd'
>
  Button
</Button>
```

```
<Button
  bg='#41d2f2'
  color='fff'
>
  Button
</Button>
```

Theme

Use theme object to define or extend theme styles

```
import { theme } from '@chakra-ui/core';

export default {
  ...theme,
  colors: {
    ...theme.colors,
    brand: {
      white: '#fefefe',
      blue: '#41d2f2',
      yellow: '#f2dc6d',
      purple: '#7700bb',
    },
  },
};
```

```
<Button bg='#41d2f2' color='brand.white' />
```

Responsive styles

Define values to each breakpoint respectively

```
.Box {
  width: 100%;
}

@media screen and (min-width: 40em) {
  width: 50%;
}

@media screen and (min-width: 52em) {
  width: 25%;
}
```

```
<Box width={['100%', 1 / 2, 1 / 4]} />
```


Q/A

Styles overlapping

Props passed by Styled System should put above the original ones.

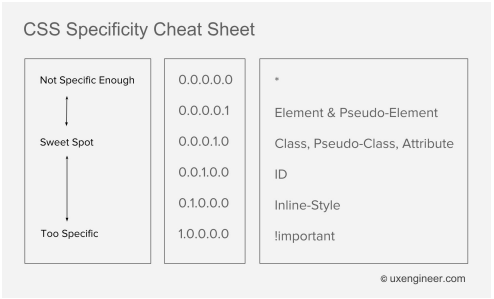
```
const Box = styled.div`
  ${color}
  margin: 15px 0;
  padding: 15px;
  border-radius: 10px;
`;
```

Avoid css overlap


```
const Box = styled.div`
  margin: 15px 0;
  padding: 15px;
  border-radius: 10px;
  ${color}
`;
```

Traditional CSS rules feat. Styled Components

```
#root div { color: red; } /* in site.css, score: 100 + 1 = 101 */

.jqouBD { color: black; } /* in styled component, score: 10 */
```

source

```
#root div { color: red; } /* in site.css, score: 100 + 1 = 101 */

.jqouBD { color: black !important; } /* in styled component, score: 10000 */
```

Class name without a hash for end-to-end testing

Use a chaninable method “attrs” to attach props to the styled component.

```
<Button>Click me!</Button>
```

```
const Button = styled.button.attrs({ className: 'button-submit' })`...`;
```


View in browser inspector

```
<button class="sc-EHOje button-submit">Click me!</button>
```

References 

- Styled System https://styled-system.com/
- We need a better UI component library - Styled System http://bit.ly/2roJEsq
- 從 Styled System 看下一代 CSS 技術 https://bit.ly/2u0xvLJ
- The Three Tenets of Styled System http://bit.ly/35ygrcV
- Styled System: Pseudo selectors in Variant http://bit.ly/35yqGhy
- How to create responsive UI with styled-components http://bit.ly/34lhxJ3
- CSS 實戰心法 http://bit.ly/2shXUn4