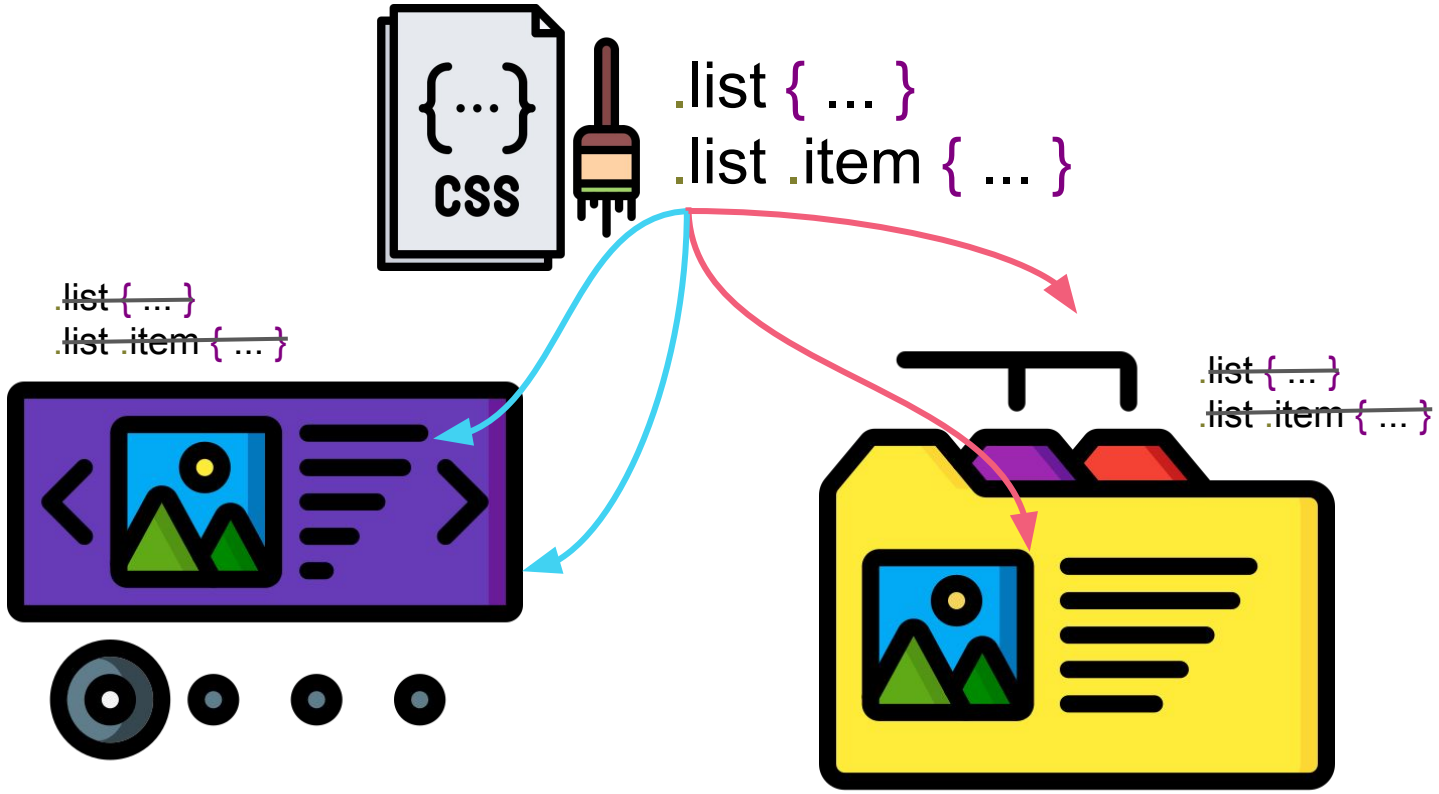
CSS

Agenda

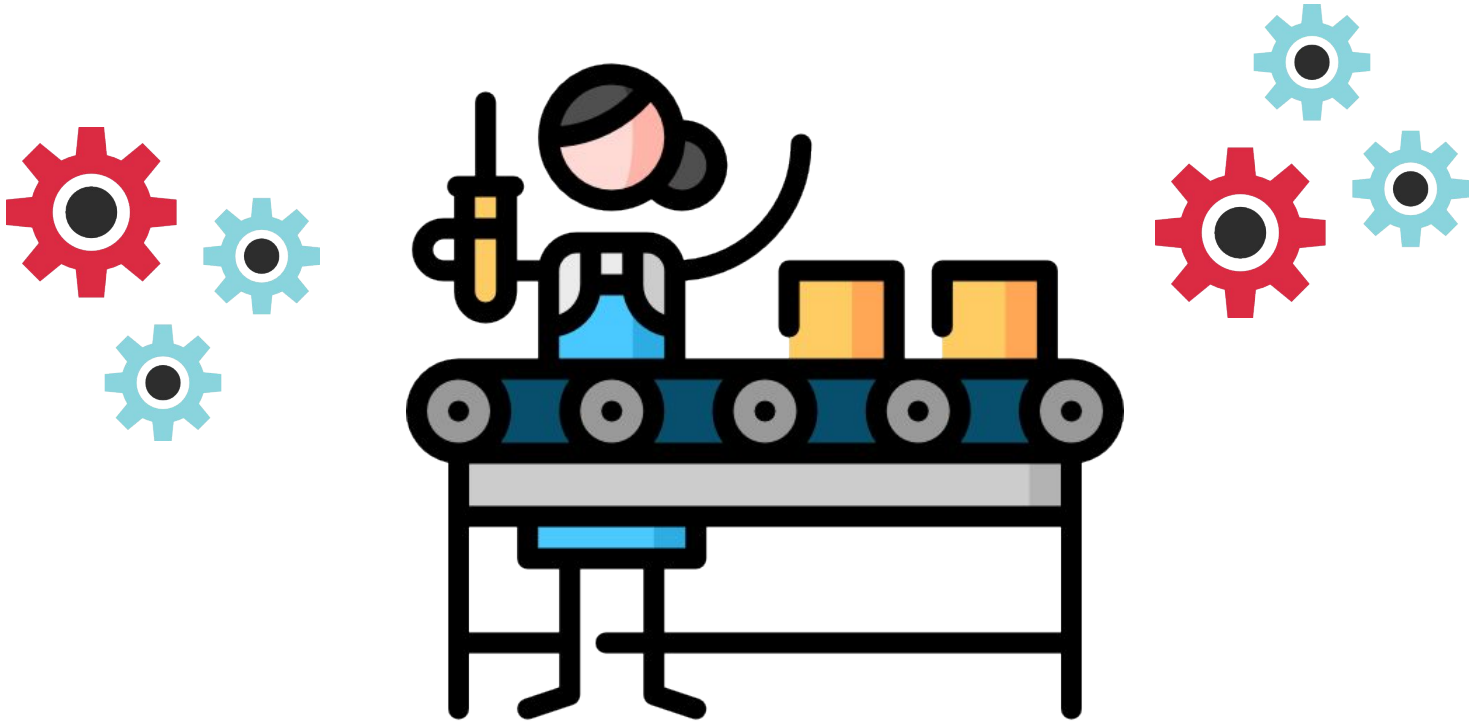
- CSS Methodologies
- Styled System
- Tonic Styled UI
- Homework: Styled Todo List with Tonic Styled UI

Everything in CSS is global.

Name Collisions



Reusability



CSS Methodologies

CSS methodologies are the solution.

- OOCSS (Object-Oriented CSS)
- BEM (Block, Element, Modifier)
- SMACSS (Scalable and Modular Architecture for CSS)
- CSS Modules
- CSS in JS

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;
    }
  }
}
```

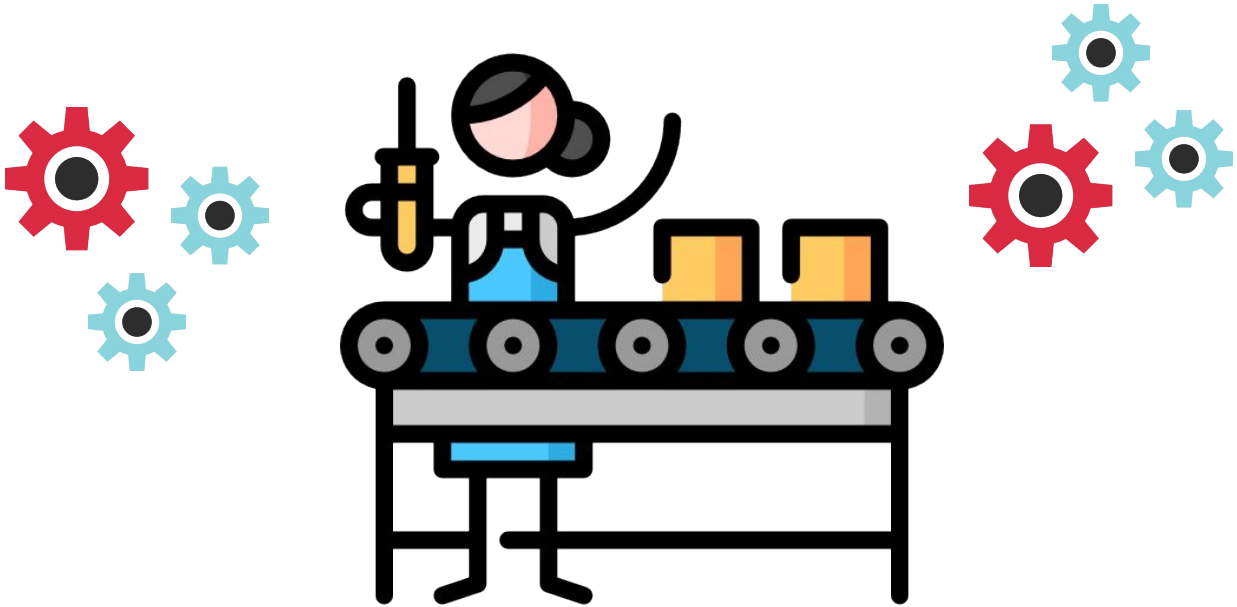
```
.search-box-mixin('page');
```

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

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

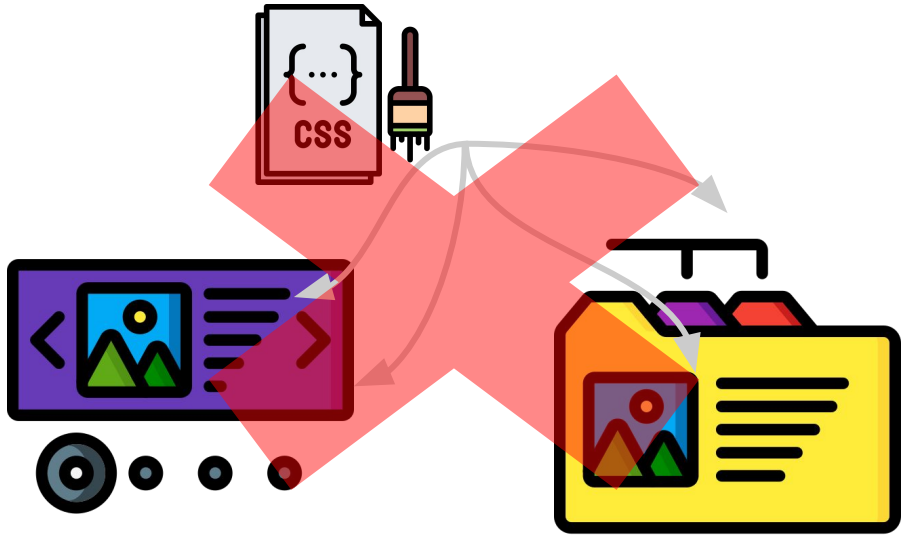
What problems do OOCSS, BEM and SMACSS solve?

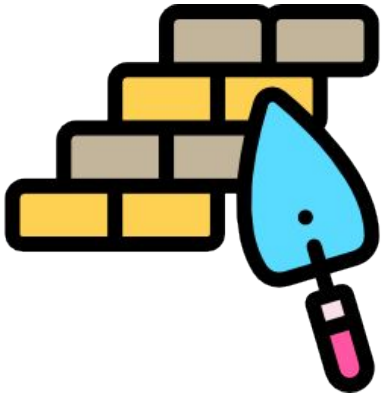
DRY stylesheets, more flexible, modular, reusable!



Global scope is still a problem!

Name collisions still happen in the global scope.

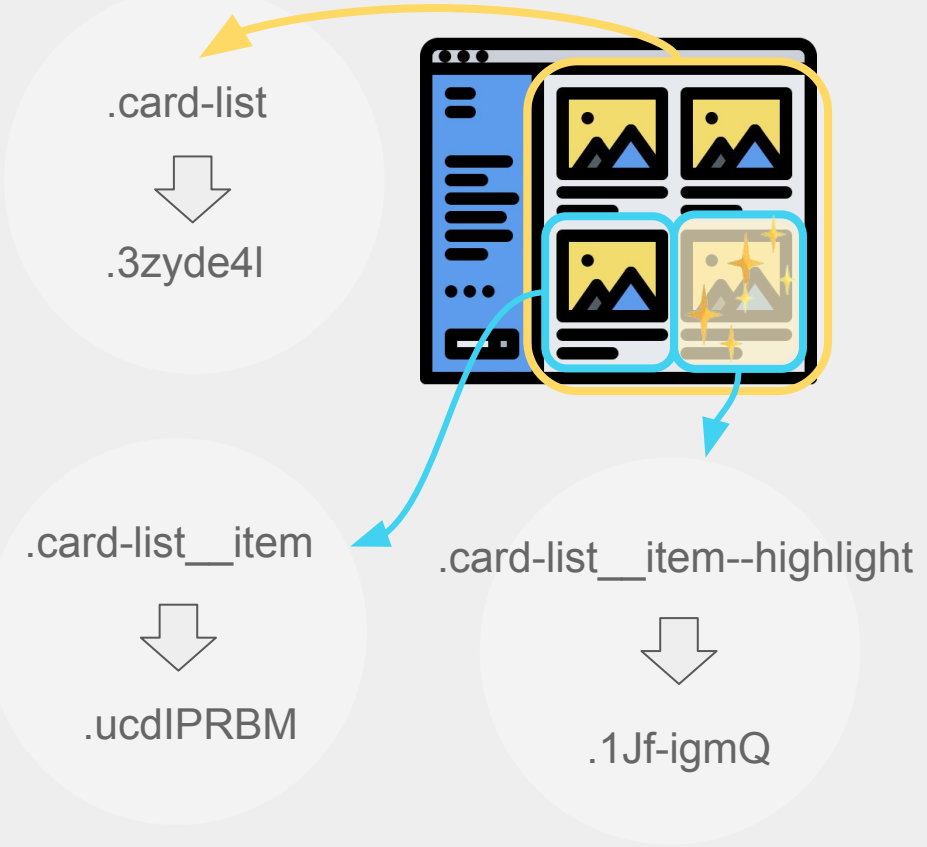


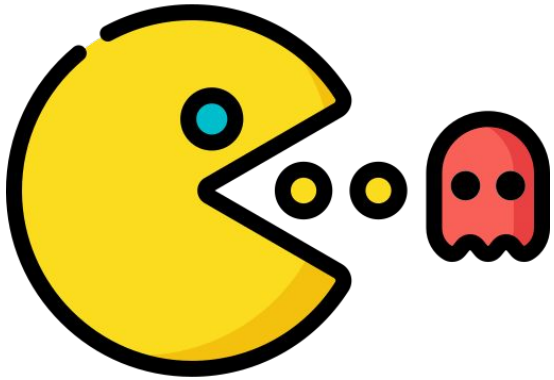


Components are the building blocks of modern web applications.

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?

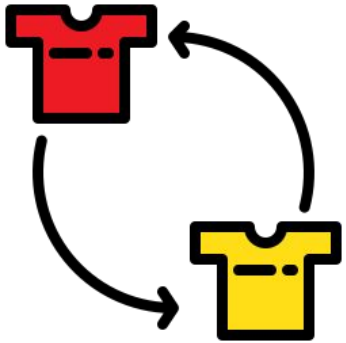
Start your app the better way with

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>
```

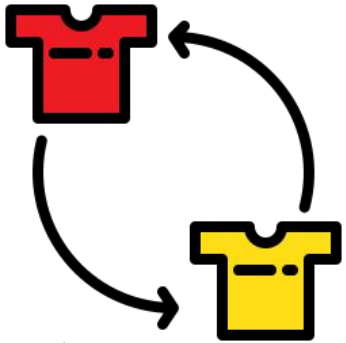
Use color utility function in Styled System

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

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

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

Easy to custom styles

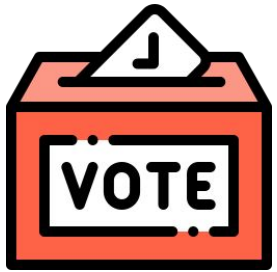


- Define global theme by a config object
- Define individual component styles by variants

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" />
```

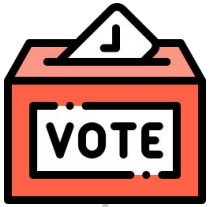
```
const buttonStyle = variant({ key:
'buttons'});
const buttonSizeStyle = variant({
prop: 'size', key: 'buttons.size' });

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;
}
```

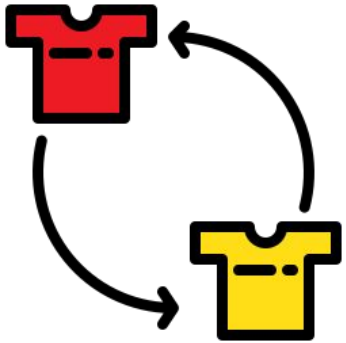
Styled System


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

Styled System is good!

More concise codes

Consistency



Easy to custom styles

Mobile-First

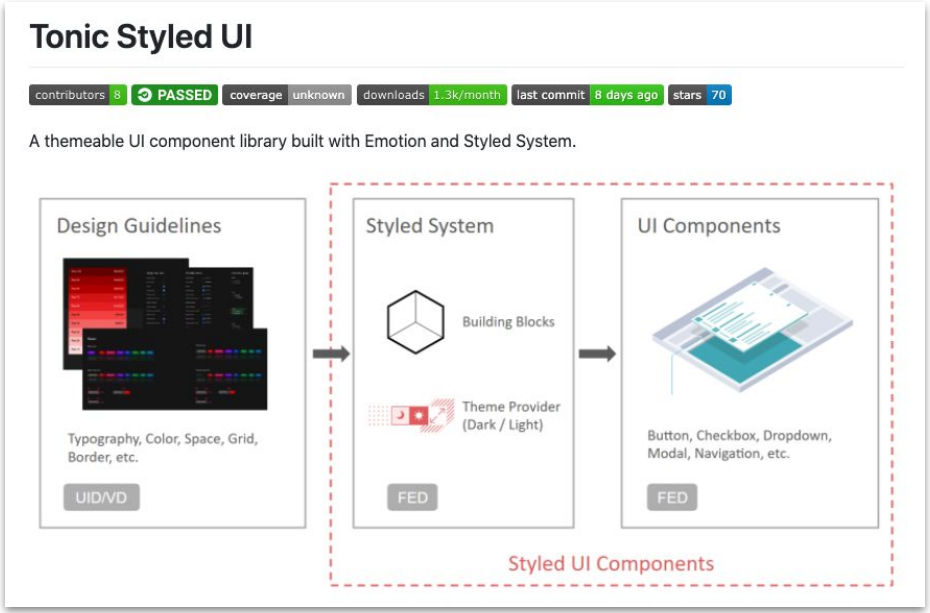
Example:
https://codesandbox.io/s/styled-system-example-forked-b7qrv

以前的我：

現在的我：

Tonic Styled UI

A themeable UI component library built with Emotion and Styled System.



https://trendmicro-frontend.github.io/styled-ui/getting-started

Styled Todo List with Tonic Styled UI

How to refactor existing code with Tonic Styled UI?

- Use <Flex> to layout
- Change <button> to <Button>
- Change <input> to <Input>

How to add more features with Tonic Styled UI?

- Add a user-defined styled-component, ex: <Button> -> <FatButton>
- Custom theme

https://codesandbox.io/s/dazzling-moore-8fu58

Should I use <Space> to layout components?

Before

After

Should I use <Space> to layout components?

```
<Flex>
  <Input value={value} onChange={change} onKeyDown={add} />
  <Space width='2x' />
  <Button variant='primary' onClick={add} onKeyDown={add}>
    Add
  </Button>
</Flex>
```

Should I use <Space> to layout components?

```
<Flex>

  <Input mr='2x' value={value} onChange={change}
onKeyDown={add} />

  <Button variant='primary' onClick={add} onKeyDown={add}>

    Add

  </Button>

 </Flex>
```

Homework 1: Styled Todo List with Tonic Styled UI

Tonic
Styled UI

?

https://codesandbox.io/s/dazzling-moore-8fu58


Q
A