README.md

CSS Modules to Tailwind CSS

This is a tool to convert css-modules to tailwind-css

Support list

- [x] tsconfig.json alias support, like `alias/component/index.module.css`
- [x] css file circular reference
- [x] project level support, just run this command: `npx css-modules-to-tailwind src/**/*.tsx`
- [ ] pseudo-classes support

About

- CSS Modules
- Tailwind CSS

Install

Global install:

```
npm install css-modules-to-tailwind -g
```

Or use `npx`:

```
npx css-modules-to-tailwind src/index.tsx
// npx css-modules-to-tailwind src/**/*.tsx
```

It will check your git directory is clean, you can use '--force' to skip the check.

How it works

It uses jscodeshift and postcss.

Try it yourself:

1. First, Create a new jsx/tsx file(index.tsx/jsx):

```jsx
import React from 'react';
import style from './index.module.css';

export const User = () => (
  <div className={style.header}>
    <div className={style.user}>
      <img className={style.avatar} alt="avatar" />
      <span className={style.username}>username</span>
    </div>
    <div className={style.channelName}>name</div>
  </div>
);
```

2. Create a new css modules file:

```css
.header {
  width: 100%;
  display: flex;
```

```
  align-items: center;
  justify-content: space-between;
}

.user {
  display: flex;
  align-items: center;
  font-weight: bold;
}

.avatar {
  width: 0.625rem;
  height: 0.625rem;
}

.username {
  font-size: 0.75rem;
  line-height: 1rem;
  color: #7DD3FC;
  margin-left: 0.25rem;
}
```

3. Use this tool now:

```
npx css-modules-to-tailwind index.tsx
```

4. You will get:

```
// index.tsx
import React from 'react';

export const User = () => (
  <div className='items-center flex justify-between w-full'>
    <div className='items-center flex font-bold'>
      <img className='h-2.5 w-2.5' alt="avatar" />
      <span className='text-sky-300 text-xs ml-1'>username</span>
    </div>
    <div className={` `}>name</div>
  </div>
);
```

> If the css file content is empty, import specifiers and css files will be removed, unused class will be replaced with ``, You should search globally for ``, then delete them.

🙋‍♂️ Flat and single structure design makes this tool work better.

Only css-modules?

Of course not. It can also be used for less/scss modules, but it doesn't work very well, like:

```
.selector1 {
  selector2();
}

.selector2 {
  font-size: 0.75rem;
  line-height: 1rem;
}
```

It just becomes:

```
.selector1 {
  selector2();
}
```

I think you should use `composes`.

Inappropriate scenes

Unreasonable nesting

```
import style form 'index.module.css';

const User = () => (
  <>
    <div className={style.parentA}>
      <div className={style.childrenA}>childrenA</div>
    </div>
    <div className={style.parentB}>
      <div className={style.childrenA}>childrenA</div>
    </div>
  </>
);
```

```
.parentA {
  .childrenA {
    // some decl
  }
}
```

You shouldn't use nesting as namespace.

You should not write multiple/conflicting declarations in a selector

```
import clsx from 'clsx';
import style form 'index.module.css';

const User = () => (
  <>
    <div className={clsx(style.cls1, style.cls2)}></div>
  </>
);



.cls1 {
  margin-left: 0.5rem;
  display: none;
}

.cls2 {
  margin-left: 0.375rem;
  display: block
}
```

Always, it will become like this:

```
const User = () => (
  <>
    <div className={clsx('hidden ml-2', 'block ml-1.5')}></div>
  </>
);
```

I mean, in tailwind, " `ml-2 ml-1.5` " === " `ml-2` ", but in your code, is the latter declaration overriding the former.

Support detail

Composes

1. Quote itself

    ```
    .class1 {
      display: flex;
    }

    .class2 {
      compose: class1
    }
    ```

it just becomes:

```
.class1 {
  @apply flex;
}

.class2 {
  composes: class1
}
```

2. Other CSS file:

```
/** index1.module.css */
.test1 {
  display: flex;
}

/** index2.module.css */
.test2 {
  composes: test1 from './index1.module.css'
}
```

`index1.module.css` will be removed, and `index2.module.css`:

```
.test2 {
  @apply flex;
}
```

Multiple states

For example:

```
.button {
  width: 1.25rem; /* 20px */
}

.box .button {
  width: 2rem; /* 32px */
}
```

It just becomes:

```
.button {
  @apply w-5; /* 20px */
}
```

```
.box .button {
  @apply w-8; /* 32px */
}
```

Classes with multiple states will not do too much processing, because I don't know if there is a conflict between the states.

Permutations

Multiple style declarations can form a Tailwind CSS class. For example:

```
.truncate {
  overflow: hidden;
  text-overflow: ellipsis;
  white-space: nowrap;
}
```

```
const Com = () => <div className={style.truncate}>text</div>
```

It will become:

```
const Com = () => <div className='truncate'>text</div>
```

Of course, it supports more complex permutations and combinations, you can try it.

Do i have to use tailwind-css?

I think it's very useful, you can try it

Releases

No releases published

Packages

No packages published

Used by 1

@backlineint / **next-css-to-tw-poc**

Contributors 2

shiyangzhaoa SSSS

dependabot[bot]

Languages

TypeScript 96.6% **JavaScript** 3.4%