"Best practices" don't actually work.

Adam Wathan

Creator of Tailwind CSS


GET THAT THING OUT OF MY FACE!
chomp
TAILWIND CSS
PY-36 FLEX
ITEMS-CENTER JUSTIFY-CENTER
RELATIVE BG-NO-REPEAT
BG-CENTER BG-COVER
chicken thoughts
imgflip.com

What can we learn from Tailwind?

Agenda

What 1. What is Tailwind CSS?

How 2. How does Tailwind work?

Why 3. Why is Tailwind good?

When 4. When to use Tailwind?

5

things we can learn from **Tailwind CSS**

This matters to you because...

Faster & Prettier
UI Dev

+

SWE Best
Practices

Agenda

What	**1. What is Tailwind CSS?**
How	2. How does Tailwind work?
Why	3. Why is Tailwind good?
When	4. When to use Tailwind?

Tailwind CSS is a popular utility-first CSS framework

What?

- Faster development How?
- Scalable for large codebases How?
- Consistent look and feel across all pages How?

Agenda

What

- Tailwind CSS is a popular utility-first CSS framework

How

2. How does Tailwind work?

Why

3. Why is Tailwind good?

When

4. When to use Tailwind?

Tailwind lets you build designs in markup

React + Tailwind CSS

navbar.tsx

```tsx
<nav>
  <div className="sm:hidden pt-1.5">
    <CollapsingMenu />
  </div>
  <ul className="hidden sm:flex align-baseline gap-4 text-xl">
    <NavLinks />
  </ul>
</nav>
```

“Traditional way”

React/HTML

navbar.tsx

```tsx
<nav>
  <div className="mobile-menu">
    <CollapsingMenu />
  </div>
  <ul className="main-menu">
    <NavLinks />
  </ul>
</nav>
```

Pure CSS

navbar_styles.css

```css
.mobile-menu {
  padding-top: var(--spacing-1_5);
}

.main-menu {
  display: none;
  vertical-align: baseline;
  column-gap: var(--spacing-4);
  font-size: var(--font-size-xl);
  line-height: var(--xl-font-line-height);
}

/* Adjust styles for when screen gets larger */
@media (min-width: 640px) {
  .mobile-menu {
    display: none;
  }

  .main-menu {
    display: flex;
  }
}
```

Tailwind Approach

navbar.tsx

```tsx
<nav>
  <div className="sm:hidden pt-1.5">
    <CollapsingMenu />
  </div>
  <ul className="hidden sm:flex align-baseline gap-4 text-xl">
    <NavLinks />
  </ul>
</nav>
```

Key Point: Tailwind lets you design within markup, which reduces context-switching and makes development faster⚡

Lesson 1

Challenge traditional practices

Tailwind defines clear naming conventions

Pure CSS

```
<button class="btn-color">Click Me</button>
<div class="custom-margin">This has margin!</div>
<section class="styled-section">Styled
Section</section>
```

Pure CSS

```
<button class="btn-color">Click Me</button>
<div class="custom-margin">This has margin!</div>
<section class="styled-section">Styled
Section</section>
```

Tailwind CSS

```
<button class="text-red-500 hover:text-red-700">Click Me</button>
<div class="my-4 px-8">This has margin and padding!</div>
<section class="bg-gray-200 rounded-lg p-4">Styled
Section</section>
```

Tailwind CSS

```
<button class="text-red-500 hover:text-red-700">Click Me</button>
<div class="my-4 px-8">This has margin and padding!</div>
<section class="bg-gray-200 rounded-lg p-4">Styled
Section</section>
```

Key Point: Tailwind defines clear naming conventions for self-documenting code 📄

Lesson 2

Use good naming conventions

Agenda

What

- Tailwind CSS is a popular utility-first CSS framework

How

- Allows designing within markup which reduces context-switching
- Defines clear naming conventions for self-documenting code

1 **Challenge traditional practices**

2 **Use good naming conventions**

Why

3. Why is Tailwind good?

When

4. When to use Tailwind?

Tailwind enables code reusability

Atomic classes

```
.border {
  border: 1px solid #dee2e6;
}

.border-t-4 {
  border-top-width: 4px;
}

.rounded-lg {
  border-radius: 0.5rem;
}
```

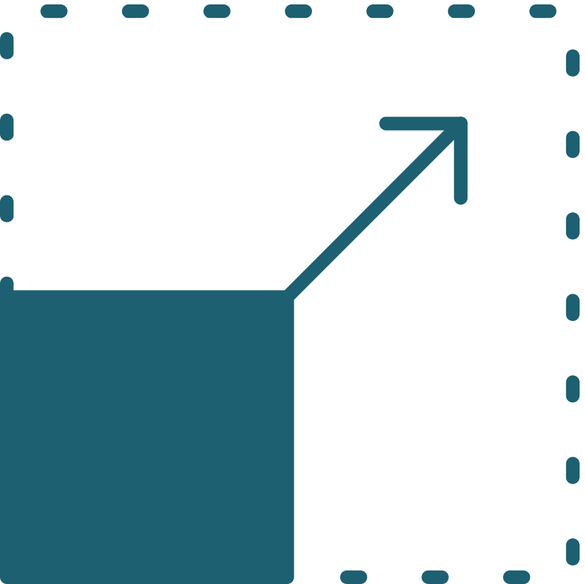
Tailwind enables code reusability

Key Point: **Atomic classes allow for code reusability and design consistency**

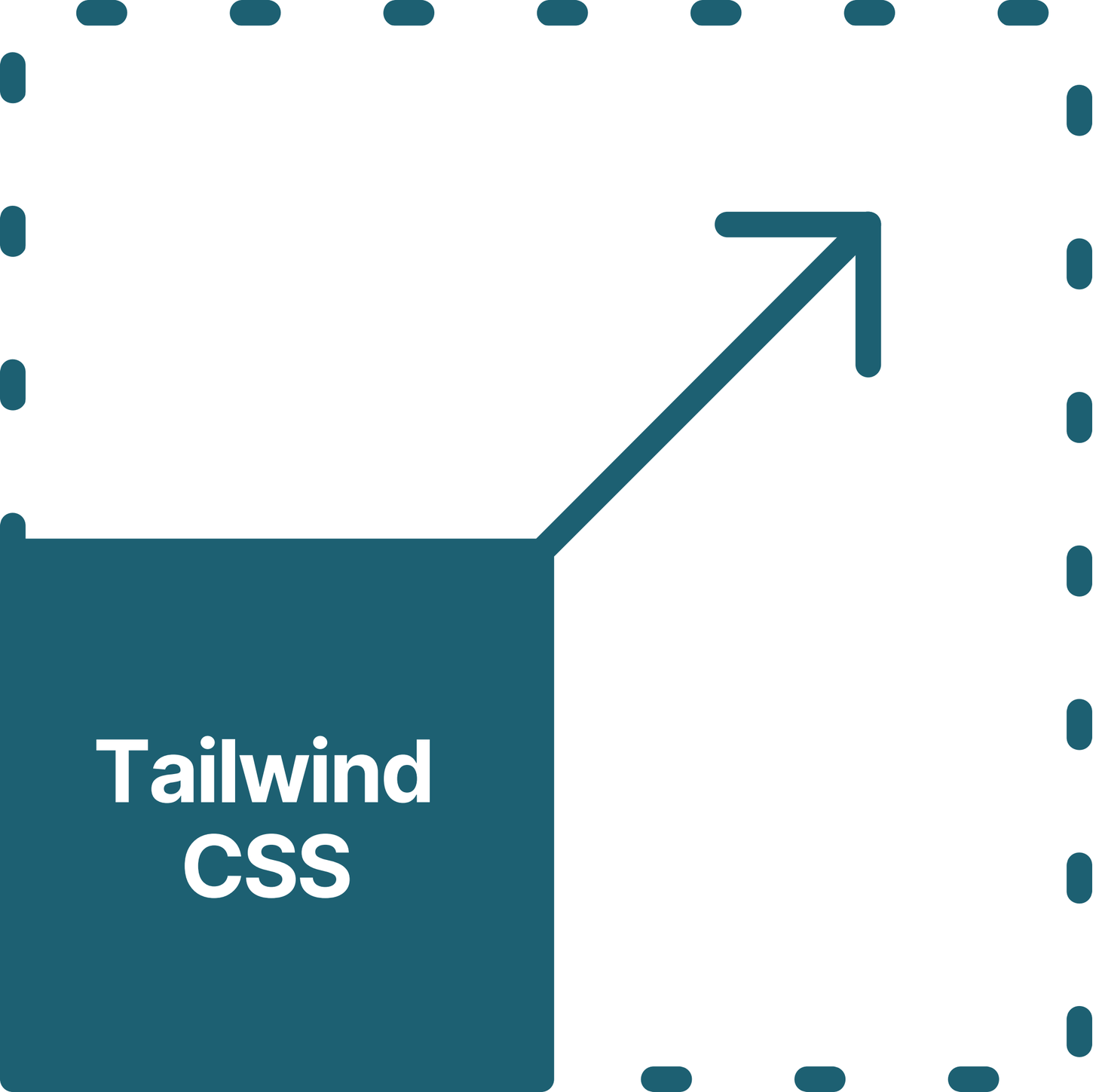
```
.border {
  border: 1px solid #dee2e6;
}

.border-t-4 {
  border-top-width: 4px;
}

.rounded-lg {
  border-radius: 0.5rem;
}
```

Lesson 3

Strive for code reusability

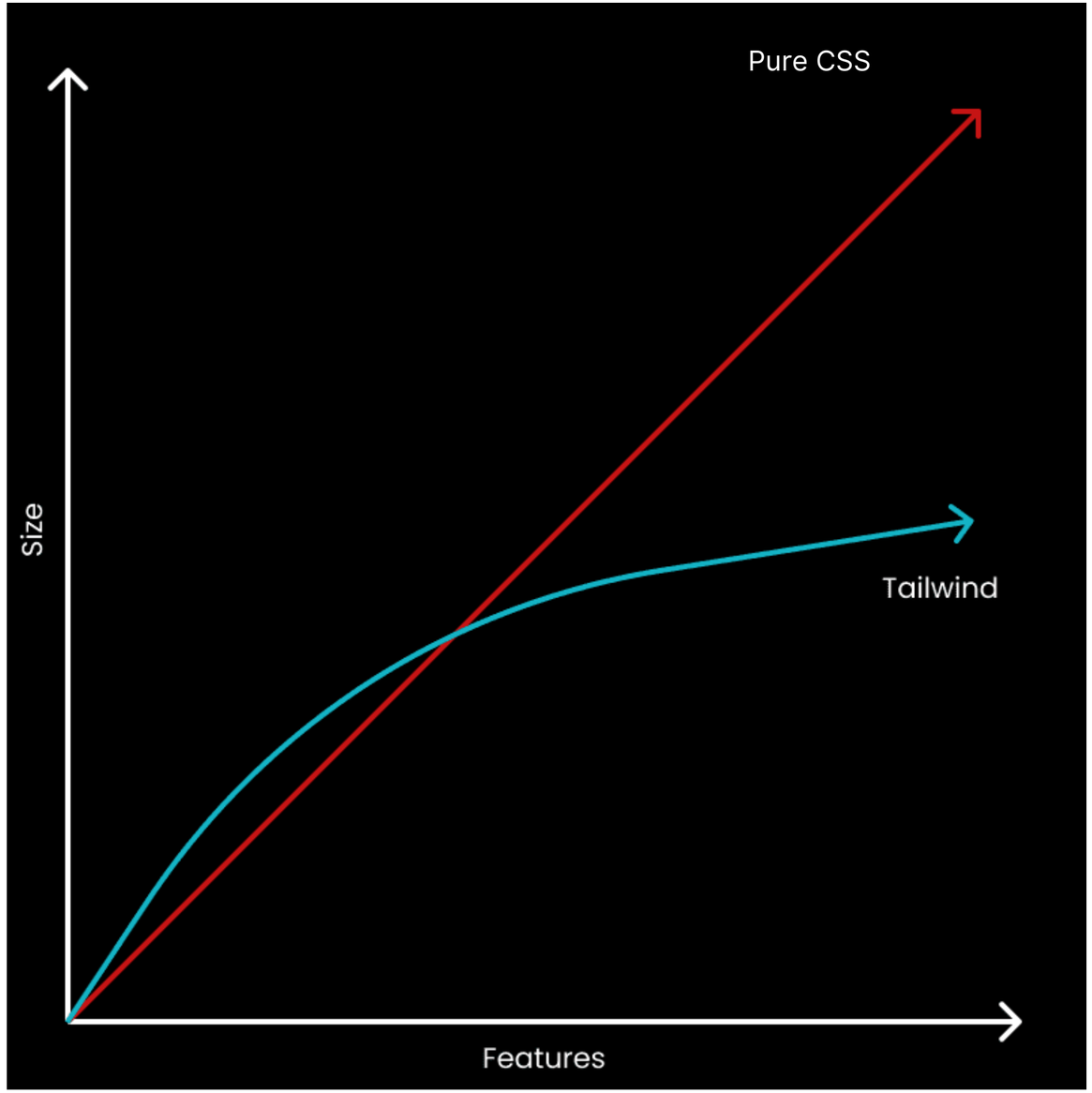


Tailwind CSS is...



Scalable!

Tailwind CSS is scalable



Tailwind CSS is scalable

Key Point: **Tailwind + PurgeCSS remove unused styles and allow scalability**

PurgeCSS

Lesson 4

Ship less code, do more

Agenda

What

- Tailwind CSS is a popular utility-first CSS framework

How

- Allows designing within markup which reduces context-switching
- Defines clear naming conventions for self-documenting code

Why

- Atomic classes for code reusability and design consistency
- Tailwind + PurgeCSS to remove unused styles and for scalability

When

4. When to use Tailwind?

1 **Challenge traditional practices**

2 **Use good naming conventions**

3 **Strive for code reusability**

4 **Ship less code, do more**

Tailwind may or may not work for you

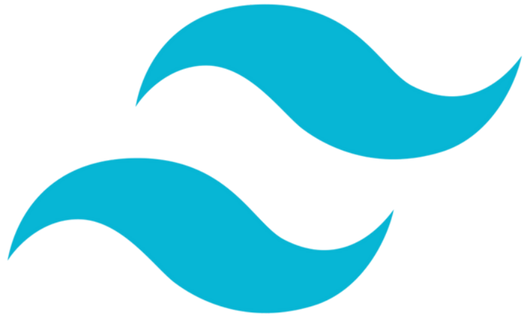
- Low context switching
- Reusability
- Scalability

- Verbose code
- Learning curve
- Animation difficulties

Which one to use?

Bootstrap

Pure CSS



Tailwind CSS

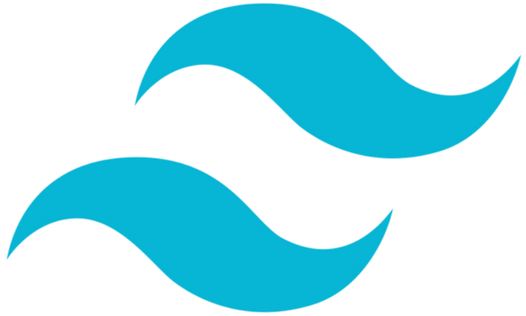
Which one to use?

Speed	Rapid Prototyping	Time-consuming	Quick
Flexibility	Pre-built components only	Maximum control	Deep customization
Maintenance	Easy until customization needed	Manual maintenance	Easy until heavy customization needed

Which one to use?

Rapid prototyping

Maximum control



Balanced

Key Point: **Tailwind restrains choices while providing enough flexibility to enable creativity**

Lesson 5

Always consider trade-offs

Challenge your assumptions and try Tailwind CSS!

Level up as a frontend engineer and get a fresh perspective on SWE best practices

Charisma Kausar
charisma.kausar@u.nus.edu

What

- Tailwind CSS is a popular utility-first CSS framework (Read more at tailwindcss.com)

How

- Allows designing within markup which reduces context-switching
- Defines clear naming conventions for self-documenting code

Why

- Atomic classes for code reusability and design consistency
- Tailwind + PurgeCSS to remove unused styles and for scalability

When

- Bootstrap for rapid prototyping
- CSS for maximum control
- Tailwind for a balance of both

1. **Challenge traditional practices**
2. **Use good naming conventions**
3. **Strive for code reusability**
4. **Ship less code, do more**
5. **Always consider trade-offs**