CSS at Scale

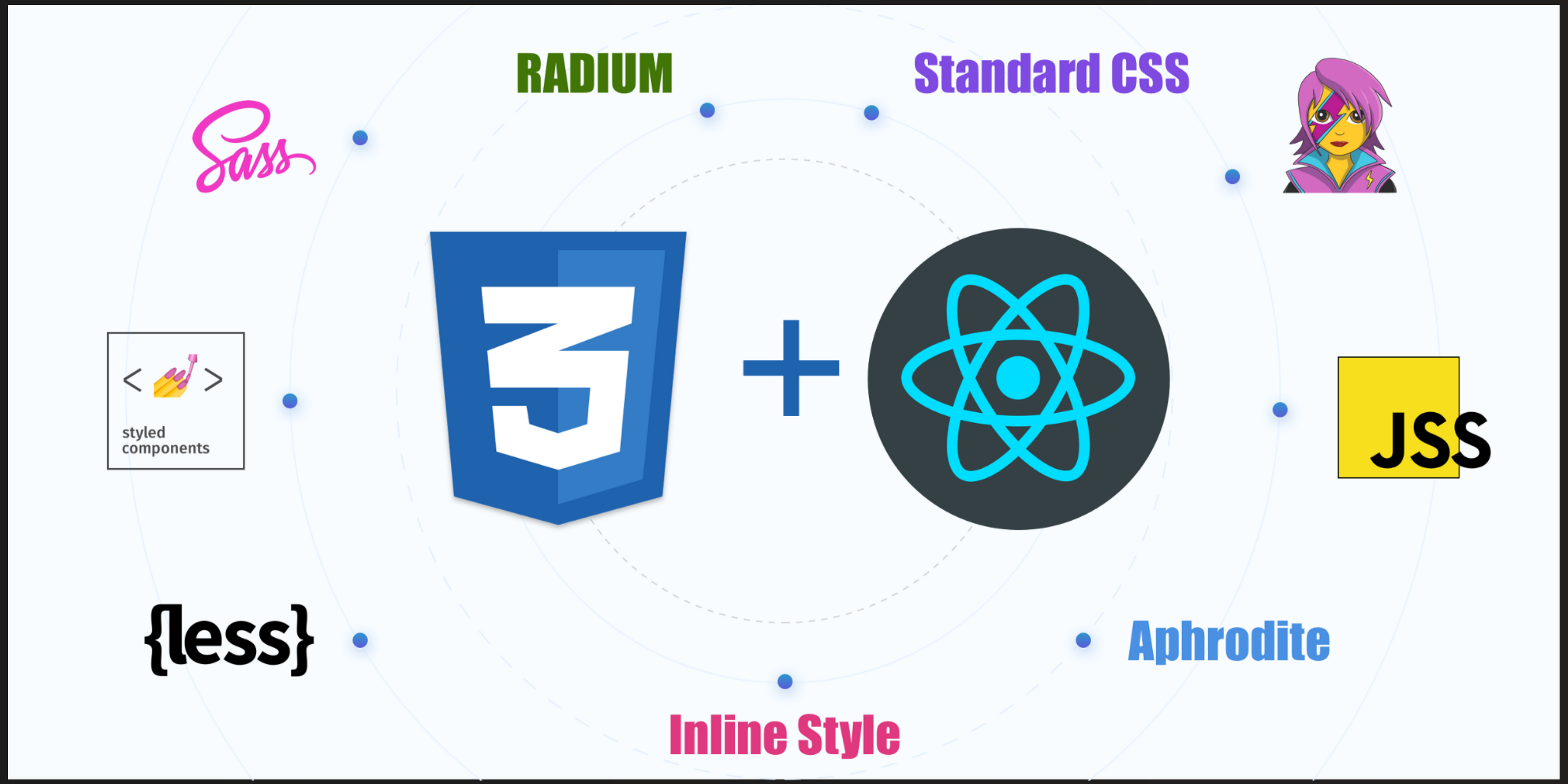
Problems with CSS, introduce tailwindCSS framework

https://rikkeisoft.com

@tult

Table of Contents

RIKKEI
where the dream begins



Common ways to style React component

Summarize some common ways to use CSS in React application briefly

Inline-style

```jsx
import React from 'react'

const Button = () => {
  return (
    <button style={{
      color: 'red',
      padding: '10px 20px',
    }}>
      Simple button
    </button>
  )
}

export default Button
```

CSS-in-CSS

```
// style.css
.button {
  color: 'red';
  padding: '10px, 20px';
  ...
}
```

```
// Button.tsx
import React from 'react'
import './style.css'


const Button = () => {
  return (
    <button className="button">Simple button</button>
  )
}

export default Button
```

CSS-in-JS

```jsx
import React from "react"
import styled from "styled-components"

const ButtonStyled = styled.button`
      color: 'red';
      padding: '10px 20px';
    `
const Button = (props) => (
  <ButtonStyled {...props}>Simple button</ButtonStyled>
```

https://rikkeisoft.com

@tult

CSS Module

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

const Button = () => {
  return (
    <button className={styles.button}>Simple button</button>
  )
}

export default Button
```

Questions

1. HOW CSS MODULE WORKS DIFFERENTLY WITH CSS-IN-CSS
2. WHAT CSS-IN-JS DOES UNDER THE HOOD?

Why we need so much approaches for only style components?

Problem with CSS-in-CSS

DEMO

Let’s look at a simple demo first.

https://codesandbox.io/s/css-seminar-hkv96?file=/src/index.tsx

Global namespace

1. All selectors are global
2. We tend to create new style instead of trying to re-used

Non-deterministic resolution

1. All selectors can be applied to any components despite the location’s of component and CSS file.

Dead code elimination

1. Cannot make sure whether a selectors only affects on only one component.

Minification

1. Because it’s hard to remove unused selectors. Which leads to harding reduce the CSS file’s size.

Sharing styling

1. Hard to reused predefined styling

Introduce TailwindCSS

Introduce CSS framework CSS and let’s see how many problems it solves

What is tailwindCSS?

TailwindCSS is a utility-first CSS framework.

- It contains a bunch of predefined classes
- Think about it like boostrap but of course... with differences.

```html
<ul class="space-y-4">
  <li>
    <div class="w-64 h-3 bg-gradient-to-br from-fuchsia-500 to-purple-600"></div>
  </li>
  <li>
    <div class="w-56 h-3 bg-gradient-to-br from-fuchsia-500 to-purple-600"></div>
  </li>
  <li>
    <div class="w-48 h-3 bg-gradient-to-br from-fuchsia-500 to-purple-600"></div>
  </li>
  <li>
    <div class="w-40 h-3 bg-gradient-to-br from-fuchsia-500 to-purple-600"></div>
  </li>
</ul>
```

Advantages of using tailwindCSS

Free in design

- Allow you to style components at low level
- Easy to customize
- Re-useability

System characteristic

- Comes with handy predefined classes
- Do not increase the CSS file size so much

Tiny in production

- Tailwind automatically removes all unusued CSS when building for production (*)
- Most tailwind projects ship less than 10kB of CSS to the client

Some notices that good to know

Adding base styles for HTML element

You can adding default style for HTML element

```css
// style.css
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer base {
  a {
    @apply text-blue-600 hover:underline;
  }
  h1 {
    @apply text-2xl font-semibold;
  }
}
```

Extracting component's style

You can extracting style of component that is used repeatedly

```
// style.css
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer components {
  .btn-blue {
    @apply px-4 py-2 rounded-lg bg-blue-500 hover:bg-blue-600;
  }
}
```

RIKKEI
where the dream begins

Extracting utilities

You also can add new utilities for reuse between many components

```
// style.css
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer utilities {
  .center-content {
    @apply absolute top-1/2 left-1/2 translate-x-1/2 -translate-y-1/2
  }
}
```

Just-in-Time mode

- Only combine the code that needed

```js
  // tailwind.config.js

module.exports = {
   mode: 'jit',
    purge: [
      // ...
    ],
    theme: {
      // ...
    }
    // ...
  }
```