NativeWind / Tailwind CSS Complete Advanced Documentation

NativeWind v4 represents a complete architectural overhaul, transforming from a simple `className` to `style` converter into a comprehensive styling system that supports the full spectrum of Tailwind CSS features. This documentation provides exhaustive coverage of all properties, advanced features, and professional implementation patterns.[1]

Complete Installation Guide

Prerequisites and Environment Setup

Before installing NativeWind, ensure your development environment meets these requirements:

- **Node.js**: 16.x or higher
- **React Native**: 0.70+ or Expo SDK 49+
- **TypeScript**: 4.9+ (recommended)
- **Metro**: Latest version for optimal bundling

Step-by-Step Installation

1. Package Installation

For **Expo SDK 50+**:

```
npx expo install nativewind tailwindcss@3.3.2 react-native-reanimated react-native-safe-a
```

For **React Native CLI**:

```
npm install nativewind tailwindcss react-native-reanimated react-native-safe-area-context
npm install --save-dev @types/react @types/react-native
```

2. Advanced Tailwind Configuration

Create a comprehensive `tailwind.config.js`:

```
const { hairlineWidth, platformSelect } = require('nativewind/theme');

/** @type {import('tailwindcss').Config} */
```

```
module.exports = {
  content: [
    "./App.{js,jsx,ts,tsx}",
    "./app/**/*.{js,jsx,ts,tsx}",
    "./components/**/*.{js,jsx,ts,tsx}",
    "./screens/**/*.{js,jsx,ts,tsx}",
    "./src/**/*.{js,jsx,ts,tsx}"
  ],
  presets: [require("nativewind/preset")],
  theme: {
    extend: {
      colors: {
        // Custom color palette with CSS variables support
        primary: {
          50: 'rgb(var(--color-primary-50) / <alpha-value>)',
          500: 'rgb(var(--color-primary-500) / <alpha-value>)',
          900: 'rgb(var(--color-primary-900) / <alpha-value>)',
        },
        // Platform-specific colors
        'platform-primary': platformSelect({
          ios: '#007AFF',
          android: '#2196F3',
          web: '#3B82F6',
          default: '#6B7280'
        }),
        // Semantic colors
        success: '#10B981',
        warning: '#F59E0B',
        error: '#EF4444',
        info: '#3B82F6'
      },
      fontFamily: {
        // Custom font definitions
        'display': ['Inter-Display', 'system-ui', 'sans-serif'],
        'body': ['Inter', 'system-ui', 'sans-serif'],
        'mono': ['JetBrains-Mono', 'Menlo', 'monospace'],
        // Platform-specific fonts
        'platform': platformSelect({
          ios: ['SF Pro Display', 'system-ui'],
          android: ['Roboto', 'system-ui'],
          web: ['Inter', 'system-ui'],
          default: ['system-ui']
        })
      },
      spacing: {
        // Custom spacing scale
        '18': '4.5rem',
        '88': '22rem',
        '128': '32rem',
        // Safe area spacing
        'safe-top': 'env(safe-area-inset-top)',
        'safe-bottom': 'env(safe-area-inset-bottom)'
      },
      borderWidth: {
        hairline: hairlineWidth()
      },
```

```
      animation: {
        // Custom animations
        'fade-in': 'fadeIn 0.5s ease-in-out',
        'slide-up': 'slideUp 0.3s ease-out',
        'bounce-in': 'bounceIn 0.6s ease-out'
      },
      keyframes: {
        fadeIn: {
          '0%': { opacity: '0' },
          '100%': { opacity: '1' }
        },
        slideUp: {
          '0%': { transform: 'translateY(100%)' },
          '100%': { transform: 'translateY(0)' }
        },
        bounceIn: {
          '0%, 20%, 40%, 60%, 80%': { transform: 'scale(0.8)' },
          '100%': { transform: 'scale(1)' }
        }
      }
    }
  },
  plugins: [
    // Add custom plugins if needed
    require('@tailwindcss/typography'),
    require('@tailwindcss/forms')
  ]
}
```

3. Global CSS with Advanced Features

Create `global.css` with comprehensive styling:

```
@tailwind base;
@tailwind components;
@tailwind utilities;

/* CSS Variables for theming */
@layer base {
  :root {
    /* Light theme variables */
    --color-primary-50: 239 246 255;
    --color-primary-500: 59 130 246;
    --color-primary-900: 30 58 138;

    /* Semantic colors */
    --color-background: 255 255 255;
    --color-surface: 249 250 251;
    --color-on-background: 17 24 39;
    --color-on-surface: 55 65 81;

    /* Elevation shadows */
    --shadow-sm: 0 1px 2px 0 rgb(0 0 0 / 0.05);
    --shadow-md: 0 4px 6px -1px rgb(0 0 0 / 0.1);
    --shadow-lg: 0 10px 15px -3px rgb(0 0 0 / 0.1);
```

```
  }

  @media (prefers-color-scheme: dark) {
    :root {
      /* Dark theme variables */
      --color-primary-50: 30 58 138;
      --color-primary-500: 147 197 253;
      --color-primary-900: 239 246 255;

      --color-background: 17 24 39;
      --color-surface: 31 41 55;
      --color-on-background: 243 244 246;
      --color-on-surface: 209 213 219;
    }
  }
}

/* Custom component classes */
@layer components {
  .card {
    @apply bg-white dark:bg-gray-800 rounded-lg shadow-md p-6;
  }

  .btn-primary {
    @apply bg-blue-500 text-white px-4 py-2 rounded-lg font-medium;
    @apply hover:bg-blue-600 active:bg-blue-700;
    @apply disabled:opacity-50 disabled:cursor-not-allowed;
  }

  .text-gradient {
    @apply bg-gradient-to-r from-blue-500 to-purple-600;
    @apply bg-clip-text text-transparent;
  }
}

/* Platform-specific utilities */
@layer utilities {
  .safe-area-top {
    padding-top: env(safe-area-inset-top);
  }

  .safe-area-bottom {
    padding-bottom: env(safe-area-inset-bottom);
  }
}
```

Complete Property Reference

[Chart 82]

Layout Properties

Property	Classes	Description	Platform Support
Display	`flex`, `hidden`	Controls element display behavior	✅ Universal
Position	`static`, `relative`, `absolute`, `fixed`	Controls positioning method	✅ Universal
Top/Right/Bottom/Left	`top-*`, `right-*`, `bottom-*`, `left-*`	Controls positioning offsets	✅ Universal
Visibility	`visible`, `invisible`, `collapse`	Controls element visibility	✅ Universal
Z-Index	`z-*` (0-50, auto)	Controls stacking order	✅ Universal
Overflow	`overflow-hidden`, `overflow-visible`, `overflow-scroll`	Controls content overflow	✔ Limited on native

Flexbox Properties

Property	Classes	Description	Notes
Flex	`flex-1`, `flex-auto`, `flex-none`, `flex-initial`	Controls flex grow/shrink behavior	Essential for RN layouts
Flex Direction	`flex-row`, `flex-col`, `flex-row-reverse`, `flex-col-reverse`	Sets main axis direction	RN default: column, Web: row
Flex Wrap	`flex-wrap`, `flex-nowrap`, `flex-wrap-reverse`	Controls line wrapping	
Justify Content	`justify-start`, `justify-end`, `justify-center`, `justify-between`, `justify-around`, `justify-evenly`	Main axis alignment	
Align Items	`items-start`, `items-end`, `items-center`, `items-baseline`, `items-stretch`	Cross axis alignment	
Align Content	`content-start`, `content-end`, `content-center`, `content-between`, `content-around`, `content-evenly`	Multi-line alignment	
Align Self	`self-auto`, `self-start`, `self-end`, `self-center`, `self-stretch`, `self-baseline`	Individual item alignment	
Order	`order-*` (1-12, first, last, none)	Controls flex item order	
Flex Basis	`basis-*`, `basis-auto`, `basis-full`	Sets initial main size	
Flex Grow	`grow`, `grow-0`	Controls growth factor	

Property	Classes	Description	Notes
Flex Shrink	`shrink`, `shrink-0`	Controls shrink factor	

Spacing Properties

Padding

```
// All padding utilities with scale
const paddingClasses = {
  all: ['p-0', 'p-px', 'p-0.5', 'p-1', 'p-1.5', 'p-2', 'p-2.5', 'p-3', 'p-3.5', 'p-4', 'p
  horizontal: ['px-*'], // Same scale
  vertical: ['py-*'],   // Same scale
  individual: ['pt-*', 'pr-*', 'pb-*', 'pl-*'] // Same scale
}
```

Margin

- Same scale as padding plus negative values: `-m-*`, `-mx-*`, `-my-*`, `-mt-*`, `-mr-*`, `-mb-*`, `-ml-*`
- Auto margins: `mx-auto`, `my-auto`, `ml-auto`, `mr-auto`

Gap (Flexbox/Grid Spacing)

- `gap-*`: Controls gap between all children
- `gap-x-*`: Controls horizontal gap
- `gap-y-*`: Controls vertical gap
- `space-x-*`: Adds margin between horizontal siblings (deprecated in v4)
- `space-y-*`: Adds margin between vertical siblings (deprecated in v4)

Sizing Properties

Width

Class	Description	Value
`w-0` to `w-96`	Fixed widths	0 to 384px
`w-auto`	Auto width	auto
`w-px`	1px width	1px
`w-0.5` to `w-3.5`	Fractional widths	2px to 14px
`w-1/2` to `w-11/12`	Fractional percentages	8.33% to 91.67%
`w-full`	Full width	100%
`w-screen`	Screen width	100vw

Class	Description	Value
`w-min`	Minimum content width	min-content
`w-max`	Maximum content width	max-content
`w-fit`	Fit content width	fit-content

Height

- Same pattern as width: `h-*`, `h-screen`, `h-full`, `h-auto`, etc.
- Additional: `h-dvh` (dynamic viewport height), `h-lvh` (large viewport height), `h-svh` (small viewport height) - Web only

Min/Max Sizing

- **Min Width**: `min-w-0`, `min-w-full`, `min-w-min`, `min-w-max`, `min-w-fit`
- **Max Width**: `max-w-none`, `max-w-xs`, `max-w-sm`, `max-w-md`, `max-w-lg`, `max-w-xl`, `max-w-2xl` to `max-w-7xl`, `max-w-full`, `max-w-min`, `max-w-max`, `max-w-fit`, `max-w-prose`, `max-w-screen-sm` to `max-w-screen-2xl`
- **Min Height**: `min-h-0`, `min-h-full`, `min-h-screen`, `min-h-min`, `min-h-max`, `min-h-fit`
- **Max Height**: `max-h-*` (same scale as height), `max-h-screen`, `max-h-min`, `max-h-max`, `max-h-fit`

Typography Properties

Font Family

```
const fontFamilies = {
  sans: 'font-sans',      // Default sans-serif stack
  serif: 'font-serif',    // Serif font stack
  mono: 'font-mono'       // Monospace font stack
}
```

Font Size with Line Height

Class	Font Size	Line Height	Rem (Web)	Rem (Native)
`text-xs`	12px	16px	0.75rem	0.857rem
`text-sm`	14px	20px	0.875rem	1rem
`text-base`	16px	24px	1rem	1.143rem
`text-lg`	18px	28px	1.125rem	1.286rem
`text-xl`	20px	28px	1.25rem	1.429rem
`text-2xl`	24px	32px	1.5rem	1.714rem
`text-3xl`	30px	36px	1.875rem	2.143rem

Class	Font Size	Line Height	Rem (Web)	Rem (Native)
`text-4xl`	36px	40px	2.25rem	2.571rem
`text-5xl`	48px	1	3rem	3.429rem
`text-6xl`	60px	1	3.75rem	4.286rem
`text-7xl`	72px	1	4.5rem	5.143rem
`text-8xl`	96px	1	6rem	6.857rem
`text-9xl`	128px	1	8rem	9.143rem

Font Weight

Class	Value	Description
`font-thin`	100	Ultra light
`font-extralight`	200	Extra light
`font-light`	300	Light
`font-normal`	400	Normal/Regular
`font-medium`	500	Medium
`font-semibold`	600	Semi bold
`font-bold`	700	Bold
`font-extrabold`	800	Extra bold
`font-black`	900	Black/Heavy

Advanced Typography

- **Letter Spacing**: `tracking-tighter`, `tracking-tight`, `tracking-normal`, `tracking-wide`, `tracking-wider`, `tracking-widest`
- **Line Height**: `leading-3` to `leading-10`, `leading-none`, `leading-tight`, `leading-snug`, `leading-normal`, `leading-relaxed`, `leading-loose`
- **Text Decoration**: `underline`, `overline`, `line-through`, `no-underline`
- **Text Transform**: `uppercase`, `lowercase`, `capitalize`, `normal-case`
- **Text Align**: `text-left`, `text-center`, `text-right`, `text-justify`
- **Vertical Align**: `align-baseline`, `align-top`, `align-middle`, `align-bottom`, `align-text-top`, `align-text-bottom`
- **White Space**: `whitespace-normal`, `whitespace-nowrap`, `whitespace-pre`, `whitespace-pre-line`, `whitespace-pre-wrap`
- **Text Overflow**: `truncate`, `text-ellipsis`, `text-clip`

Color Properties

Text Colors

Full Tailwind color palette support:

- **Grays**: `text-slate-*`, `text-gray-*`, `text-zinc-*`, `text-neutral-*`, `text-stone-*`
- **Colors**: `text-red-*`, `text-orange-*`, `text-amber-*`, `text-yellow-*`, `text-lime-*`, `text-green-*`, `text-emerald-*`, `text-teal-*`, `text-cyan-*`, `text-sky-*`, `text-blue-*`, `text-indigo-*`, `text-violet-*`, `text-purple-*`, `text-fuchsia-*`, `text-pink-*`, `text-rose-*`
- **Shades**: Each color has shades from 50 (lightest) to 950 (darkest)

Background Colors

- Same color system as text: `bg-*`
- Gradients (Web only): `bg-gradient-to-r`, `from-*`, `via-*`, `to-*`

Border Colors

- Same color system: `border-*`

Border Properties

Border Width

- `border` (1px), `border-0`, `border-2`, `border-4`, `border-8`
- Directional: `border-t-*`, `border-r-*`, `border-b-*`, `border-l-*`
- Individual corners: `border-s-*`, `border-e-*` (logical properties)

Border Radius

Class	Value	Description
`rounded-none`	0	No rounding
`rounded-sm`	2px	Small rounding
`rounded`	4px	Default rounding
`rounded-md`	6px	Medium rounding
`rounded-lg`	8px	Large rounding
`rounded-xl`	12px	Extra large
`rounded-2xl`	16px	2X large
`rounded-3xl`	24px	3X large
`rounded-full`	9999px	Fully rounded

Directional radius:

- `rounded-t-*`, `rounded-r-*`, `rounded-b-*`, `rounded-l-*`
- `rounded-tl-*`, `rounded-tr-*`, `rounded-bl-*`, `rounded-br-*`

Border Style

- `border-solid`, `border-dashed`, `border-dotted`, `border-double`, `border-none`
- **Note**: Limited support on React Native

Effects and Transforms

Box Shadow

Class	Value
`shadow-sm`	0 1px 2px 0 rgb(0 0 0 / 0.05)
`shadow`	0 1px 3px 0 rgb(0 0 0 / 0.1), 0 1px 2px -1px rgb(0 0 0 / 0.1)
`shadow-md`	0 4px 6px -1px rgb(0 0 0 / 0.1), 0 2px 4px -2px rgb(0 0 0 / 0.1)
`shadow-lg`	0 10px 15px -3px rgb(0 0 0 / 0.1), 0 4px 6px -4px rgb(0 0 0 / 0.1)
`shadow-xl`	0 20px 25px -5px rgb(0 0 0 / 0.1), 0 8px 10px -6px rgb(0 0 0 / 0.1)
`shadow-2xl`	0 25px 50px -12px rgb(0 0 0 / 0.25)
`shadow-inner`	inset 0 2px 4px 0 rgb(0 0 0 / 0.05)
`shadow-none`	`none`

Opacity

- `opacity-0` to `opacity-100` in increments of 5
- Special values: `opacity-5`, `opacity-10`, `opacity-20`, `opacity-25`, `opacity-30`, `opacity-40`, `opacity-50`, `opacity-60`, `opacity-70`, `opacity-75`, `opacity-80`, `opacity-90`, `opacity-95`

Transform Properties

Scale:

- `scale-0` to `scale-150` (0% to 150%)
- `scale-x-*`, `scale-y-*` for axis-specific scaling

Rotate:

- `rotate-0`, `rotate-1`, `rotate-2`, `rotate-3`, `rotate-6`, `rotate-12`, `rotate-45`, `rotate-90`, `rotate-180`
- Negative values: `-rotate-*`

Translate:

- Same scale as spacing: `translate-x-*`, `translate-y-*`
- Percentages: `translate-x-1/2`, `translate-y-full`
- Negative values supported

Skew:

- `skew-x-*`, `skew-y-*` with values: 0, 1, 2, 3, 6, 12 degrees

Transform Origin:

- `origin-center`, `origin-top`, `origin-top-right`, `origin-right`, `origin-bottom-right`, `origin-bottom`, `origin-bottom-left`, `origin-left`, `origin-top-left`

Advanced Features and Capabilities

CSS Variables Integration

NativeWind v4 provides comprehensive CSS variables support:[1]

```
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      colors: {
        // RGB space for better alpha support
        primary: 'rgb(var(--color-primary) / <alpha-value>)',
        secondary: 'rgb(var(--color-secondary) / <alpha-value>)',

        // HSL space for better color manipulation
        accent: 'hsl(var(--color-accent-hsl) / <alpha-value>)'
      },
      spacing: {
        'dynamic': 'var(--spacing-dynamic)'
      }
    }
  }
}
```

```
/* Dynamic theming in global.css */
:root {
  --color-primary: 59 130 246;
  --color-secondary: 16 185 129;
  --color-accent-hsl: 262 83% 58%;
  --spacing-dynamic: 1rem;
}

@media (prefers-color-scheme: dark) {
  :root {
    --color-primary: 147 197 253;
    --color-secondary: 52 211 153;
    --spacing-dynamic: 1.5rem;
```

```
  }
}
```

Platform-Specific Styling

Platform Variants

Variant	Target Platform(s)
`ios:`	iOS only
`android:`	Android only
`web:`	Web only
`windows:`	Windows only
`osx:`	macOS only
`native:`	All native platforms (iOS, Android, Windows, macOS)

```
<View className="
  bg-white
  ios:bg-gray-100
  android:bg-blue-50
  web:bg-gradient-to-br web:from-blue-50 web:to-indigo-100
  native:shadow-lg
  web:shadow-xl
">
  <Text className="
    text-lg
    ios:font-semibold
    android:font-medium
    web:font-bold
    native:text-gray-900
    web:text-blue-900
  ">
    Platform-specific styling
  </Text>
</View>
```

Advanced Platform Configuration

```
// tailwind.config.js - Advanced platform theming
const { platformSelect } = require('nativewind/theme');

module.exports = {
  theme: {
    extend: {
      colors: {
        'adaptive-primary': platformSelect({
          ios: '#007AFF',      // iOS system blue
          android: '#2196F3',  // Material Design blue
```

```
          web: '#3B82F6',      // Tailwind blue-500
          default: '#6B7280'   // Fallback gray
        }),
        'platform-surface': platformSelect({
          ios: '#F2F2F7',      // iOS grouped background
          android: '#FFFFFF',  // Material surface
          web: '#F9FAFB'       // Web light gray
        })
      },
      fontFamily: {
        'system': platformSelect({
          ios: ['SF Pro Text', '-apple-system', 'BlinkMacSystemFont'],
          android: ['Roboto', 'system-ui'],
          web: ['Inter', 'system-ui', 'sans-serif'],
          default: ['system-ui', 'sans-serif']
        })
      },
      spacing: {
        'safe-area': platformSelect({
          ios: '44px',     // iOS safe area
          android: '24px', // Android status bar
          web: '0px'       // No safe area on web
        })
      }
    }
  }
}
```

State-Based Styling

Pseudo-Class Support

Pseudo-class	Supported Components	Description
`hover:`	Pressable, TouchableOpacity, Button	Mouse hover state
`focus:`	TextInput, Pressable	Focus state
`active:`	Pressable, TouchableOpacity	Press/touch state
`disabled:`	Pressable, Button, TextInput	Disabled state
`pressed:`	Pressable	Pressed state (alias for active)

```
<Pressable className="
  bg-blue-500
  hover:bg-blue-600
  active:bg-blue-700
  disabled:bg-gray-300
  disabled:opacity-50
  transition-colors duration-200
">
  <Text className="
    text-white
    group-hover:text-blue-100
```

```
    group-active:text-blue-200
  ">
    Interactive Button
  </Text>
</Pressable>
```

Group and Parent State Modifiers

```
<View className="group">
  <Pressable className="
    bg-white
    group-hover:bg-gray-50
    group-active:bg-gray-100
    p-4 rounded-lg
  ">
    <View className="flex-row items-center">
      <View className="
        w-3 h-3 bg-gray-400 rounded-full
        group-hover:bg-blue-500
        group-active:bg-blue-600
        transition-colors
      " />
      <Text className="
        ml-3 text-gray-900
        group-hover:text-blue-900
        group-active:text-blue-950
      ">
        Hover me to see group effects
      </Text>
    </View>
  </Pressable>
</View>
```

Responsive Design System

Breakpoints

Breakpoint	Min Width	Typical Usage
`sm:`	640px	Small tablets
`md:`	768px	Tablets
`lg:`	1024px	Small desktops
`xl:`	1280px	Large desktops
`2xl:`	1536px	Extra large screens

```
<View className="
  w-full
  sm:w-1/2
  md:w-1/3
```

```
  lg:w-1/4
  xl:w-1/5
  p-4
  sm:p-6
  lg:p-8
">
  <Text className="
    text-base
    sm:text-lg
    md:text-xl
    lg:text-2xl
    font-normal
    md:font-medium
    lg:font-semibold
  ">
    Responsive text
  </Text>
</View>
```

Container Queries

```
<View className="@container">
  <View className="
    grid grid-cols-1
    @sm:grid-cols-2
    @lg:grid-cols-3
    gap-4
  ">
    <Text className="
      text-sm
      @md:text-base
      @lg:text-lg
    ">
      Container query responsive text
    </Text>
  </View>
</View>
```

Animation System (Experimental)

NativeWind v4 provides experimental animation support through React Native Reanimated:[2]

Built-in Animations

```
<View className="
  animate-pulse
  bg-gray-200
  rounded-lg
  h-20
">
  <View className="animate-spin w-6 h-6 border-2 border-blue-500 border-t-transparent rou
</View>
```

```
<Text className="animate-bounce text-2xl">
  Bouncing text
</Text>

<View className="animate-ping absolute w-4 h-4 bg-red-500 rounded-full opacity-75" />
```

Custom Keyframe Animations

```
/* In global.css */
@keyframes fadeInUp {
  0% {
    opacity: 0;
    transform: translateY(20px);
  }
  100% {
    opacity: 1;
    transform: translateY(0);
  }
}

@keyframes scaleIn {
  0% {
    transform: scale(0.8);
    opacity: 0;
  }
  100% {
    transform: scale(1);
    opacity: 1;
  }
}
```

```
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      animation: {
        'fade-in-up': 'fadeInUp 0.6s ease-out',
        'scale-in': 'scaleIn 0.3s ease-out'
      }
    }
  }
}
```

```
<View className="animate-fade-in-up">
  <Text className="animate-scale-in delay-200">
    Custom animated content
  </Text>
</View>
```

Dark Mode Implementation

System-Based Dark Mode

```
import { useColorScheme } from 'nativewind';

export default function ThemedScreen() {
  const { colorScheme, setColorScheme, toggleColorScheme } = useColorScheme();

  return (
    <View className="
      flex-1
      bg-white dark:bg-gray-900
      transition-colors duration-300
    ">
      <Text className="
        text-gray-900 dark:text-gray-100
        text-lg font-medium
      ">
        Current theme: {colorScheme}
      </Text>

      <Pressable
        onPress={toggleColorScheme}
        className="
          mt-4 px-6 py-3 rounded-lg
          bg-blue-500 dark:bg-blue-600
          active:bg-blue-600 dark:active:bg-blue-700
        "
      >
        <Text className="text-white font-medium text-center">
          Toggle Theme
        </Text>
      </Pressable>
    </View>
  );
}
```

Advanced Dark Mode with CSS Variables

```
/* global.css */
@layer base {
  :root {
    --color-background: 255 255 255;
    --color-foreground: 17 24 39;
    --color-card: 255 255 255;
    --color-card-foreground: 17 24 39;
    --color-border: 229 231 235;
    --color-input: 229 231 235;
  }

  .dark {
    --color-background: 17 24 39;
```

```
    --color-foreground: 243 244 246;
    --color-card: 31 41 55;
    --color-card-foreground: 243 244 246;
    --color-border: 75 85 99;
    --color-input: 75 85 99;
  }
}
```

```
// tailwind.config.js
module.exports = {
  darkMode: 'class',
  theme: {
    extend: {
      colors: {
        background: 'rgb(var(--color-background) / <alpha-value>)',
        foreground: 'rgb(var(--color-foreground) / <alpha-value>)',
        card: 'rgb(var(--color-card) / <alpha-value>)',
        'card-foreground': 'rgb(var(--color-card-foreground) / <alpha-value>)',
        border: 'rgb(var(--color-border) / <alpha-value>)',
        input: 'rgb(var(--color-input) / <alpha-value>)'
      }
    }
  }
}
```

Arbitrary Value System

Arbitrary Properties

```
<View className="
  w-[32px]
  h-[calc(100vh-64px)]
  bg-[#ff6b6b]
  shadow-[0_35px_60px_-15px_rgba(0,0,0,0.3)]
  top-[117px]
  left-[50%]
  translate-x-[-50%]
">
  <Text className="
    text-[14px]
    leading-[1.2]
    tracking-[0.5px]
    text-[hsl(280_100%_70%)]
  ">
    Arbitrary values
  </Text>
</View>
```

Arbitrary Variants

```
<View className="
  [&>*]:mb-4
  [&>*:last-child]:mb-0
  [&>.special]:bg-blue-100
">
  <Text>First child</Text>
  <Text>Second child</Text>
  <Text className="special">Special child</Text>
  <Text>Last child (no margin)</Text>
</View>
```

Performance Optimization

Static vs Dynamic Styles

```
// ✅ Good: Static styles (compiled at build time)
<View className="bg-blue-500 p-4 rounded-lg shadow-md">
  <Text className="text-white font-semibold">Static styling</Text>
</View>

// ❌ Avoid: Dynamic class construction (runtime overhead)
const getButtonColor = (variant) => `bg-${variant}-500`;
<Pressable className={getButtonColor('blue')}>...</Pressable>

// ✅ Better: Pre-defined class mappings
const buttonVariants = {
  primary: 'bg-blue-500 text-white',
  secondary: 'bg-gray-200 text-gray-900',
  success: 'bg-green-500 text-white'
};
<Pressable className={buttonVariants[variant]}>...</Pressable>
```

Tree Shaking and Content Configuration

```
// tailwind.config.js - Optimized content scanning
module.exports = {
  content: [
    // Be specific about file patterns
    './app/**/*.{js,ts,jsx,tsx}',
    './components/**/*.{js,ts,jsx,tsx}',
    './screens/**/*.{js,ts,jsx,tsx}',

    // Exclude unnecessary files
    '!./node_modules/**/*',
    '!./**/*.test.{js,ts,jsx,tsx}',
    '!./**/*.spec.{js,ts,jsx,tsx}'
  ],

  // Enable JIT mode for faster builds
```

```
  mode: 'jit',

  // Remove unused styles in production
  purge: {
    enabled: process.env.NODE_ENV === 'production'
  }
}
```

Professional Development Patterns

Component Architecture

Compound Components with NativeWind

```
interface CardProps {
  className?: string;
  children: React.ReactNode;
}

interface CardHeaderProps {
  className?: string;
  children: React.ReactNode;
}

interface CardContentProps {
  className?: string;
  children: React.ReactNode;
}

interface CardFooterProps {
  className?: string;
  children: React.ReactNode;
}

const Card = ({ className = '', children }: CardProps) => (
  <View className={`
    bg-white dark:bg-gray-800
    rounded-lg shadow-md border
    border-gray-200 dark:border-gray-700
    ${className}
  `}>
    {children}
  </View>
);

const CardHeader = ({ className = '', children }: CardHeaderProps) => (
  <View className={`p-6 pb-2 ${className}`}>
    {children}
  </View>
);

const CardContent = ({ className = '', children }: CardContentProps) => (
  <View className={`p-6 pt-0 ${className}`}>
```

```
    {children}
  </View>
);

const CardFooter = ({ className = '', children }: CardFooterProps) => (
  <View className={`p-6 pt-2 flex-row justify-end ${className}`}>
    {children}
  </View>
);

Card.Header = CardHeader;
Card.Content = CardContent;
Card.Footer = CardFooter;

// Usage
<Card className="mx-4 my-2">
  <Card.Header>
    <Text className="text-xl font-bold text-gray-900 dark:text-gray-100">
      Profile Settings
    </Text>
  </Card.Header>
  <Card.Content>
    <Text className="text-gray-600 dark:text-gray-400">
      Manage your account preferences and settings.
    </Text>
  </Card.Content>
  <Card.Footer>
    <Button variant="outline" className="mr-2">Cancel</Button>
    <Button variant="primary">Save Changes</Button>
  </Card.Footer>
</Card>
```

Polymorphic Components

```
interface ButtonProps<T extends React.ElementType = 'button'> {
  as?: T;
  variant?: 'primary' | 'secondary' | 'outline' | 'ghost';
  size?: 'sm' | 'md' | 'lg';
  className?: string;
  children: React.ReactNode;
}

const Button = <T extends React.ElementType = 'button'>({
  as,
  variant = 'primary',
  size = 'md',
  className = '',
  children,
  ...props
}: ButtonProps<T> & Omit<React.ComponentProps<T>, keyof ButtonProps<T>>) => {
  const Component = as || Pressable;

  const baseClasses = 'items-center justify-center rounded-lg font-medium transition-colo

  const variantClasses = {
```

```
    primary: 'bg-blue-500 text-white hover:bg-blue-600 active:bg-blue-700',
    secondary: 'bg-gray-200 text-gray-900 hover:bg-gray-300 active:bg-gray-400',
    outline: 'border border-gray-300 text-gray-700 hover:bg-gray-50 active:bg-gray-100',
    ghost: 'text-gray-700 hover:bg-gray-100 active:bg-gray-200'
  };

  const sizeClasses = {
    sm: 'px-3 py-1.5 text-sm',
    md: 'px-4 py-2 text-base',
    lg: 'px-6 py-3 text-lg'
  };

  const classes = `${baseClasses} ${variantClasses[variant]} ${sizeClasses[size]} ${class

  return (
    <Component className={classes} {...props}>
      {typeof children === 'string' ? <Text>{children}</Text> : children}
    </Component>
  );
};

// Usage examples
<Button>Default Button</Button>
<Button as={TouchableOpacity} variant="outline">TouchableOpacity Button</Button>
<Button as={View} variant="ghost" size="lg">Non-interactive Element</Button>
```

Design System Implementation

Design Tokens

```
// design-tokens.js
export const designTokens = {
  colors: {
    brand: {
      primary: {
        50: '#eff6ff',
        500: '#3b82f6',
        900: '#1e3a8a'
      },
      secondary: {
        50: '#f0fdf4',
        500: '#22c55e',
        900: '#14532d'
      }
    },
    semantic: {
      success: '#10b981',
      warning: '#f59e0b',
      error: '#ef4444',
      info: '#3b82f6'
    },
    neutral: {
      50: '#f9fafb',
      100: '#f3f4f6',
```

```
      500: '#6b7280',
      900: '#111827'
    }
  },
  spacing: {
    xs: '0.5rem',    // 8px
    sm: '0.75rem',   // 12px
    md: '1rem',      // 16px
    lg: '1.5rem',    // 24px
    xl: '2rem',      // 32px
    '2xl': '3rem'    // 48px
  },
  typography: {
    fontSizes: {
      xs: '0.75rem',   // 12px
      sm: '0.875rem',  // 14px
      base: '1rem',    // 16px
      lg: '1.125rem',  // 18px
      xl: '1.25rem',   // 20px
      '2xl': '1.5rem'  // 24px
    },
    fontWeights: {
      normal: '400',
      medium: '500',
      semibold: '600',
      bold: '700'
    },
    lineHeights: {
      tight: '1.25',
      normal: '1.5',
      relaxed: '1.75'
    }
  },
  radii: {
    sm: '0.125rem',  // 2px
    md: '0.375rem',  // 6px
    lg: '0.5rem',    // 8px
    xl: '0.75rem',   // 12px
    full: '9999px'
  },
  shadows: {
    sm: '0 1px 2px 0 rgb(0 0 0 / 0.05)',
    md: '0 4px 6px -1px rgb(0 0 0 / 0.1)',
    lg: '0 10px 15px -3px rgb(0 0 0 / 0.1)',
    xl: '0 20px 25px -5px rgb(0 0 0 / 0.1)'
  }
};
```

Theme Provider

```
interface ThemeContextType {
  theme: 'light' | 'dark';
  toggleTheme: () => void;
  colors: typeof designTokens.colors;
  spacing: typeof designTokens.spacing;
```

```
}

const ThemeContext = createContext<ThemeContextType | undefined>(undefined);

export const ThemeProvider: React.FC<{ children: React.ReactNode }> = ({ children }) => {
  const { colorScheme, setColorScheme } = useColorScheme();

  const toggleTheme = useCallback(() => {
    setColorScheme(colorScheme === 'light' ? 'dark' : 'light');
  }, [colorScheme, setColorScheme]);

  const value = useMemo(() => ({
    theme: colorScheme ?? 'light',
    toggleTheme,
    colors: designTokens.colors,
    spacing: designTokens.spacing
  }), [colorScheme, toggleTheme]);

  return (
    <ThemeContext.Provider value={value}>
      {children}
    </ThemeContext.Provider>
  );
};

export const useTheme = () => {
  const context = useContext(ThemeContext);
  if (!context) {
    throw new Error('useTheme must be used within a ThemeProvider');
  }
  return context;
};
```

Advanced Styling Patterns

Variant-Based Styling with CVA

```
import { cva, type VariantProps } from 'class-variance-authority';
import { cn } from '../utils/cn';

const buttonVariants = cva(
  'inline-flex items-center justify-center rounded-md text-sm font-medium transition-colo
  {
    variants: {
      variant: {
        default: 'bg-primary text-primary-foreground hover:bg-primary/90',
        destructive: 'bg-destructive text-destructive-foreground hover:bg-destructive/90'
        outline: 'border border-input bg-background hover:bg-accent hover:text-accent-for
        secondary: 'bg-secondary text-secondary-foreground hover:bg-secondary/80',
        ghost: 'hover:bg-accent hover:text-accent-foreground',
        link: 'text-primary underline-offset-4 hover:underline'
      },
      size: {
        default: 'h-10 px-4 py-2',
```

```
        sm: 'h-9 rounded-md px-3',
        lg: 'h-11 rounded-md px-8',
        icon: 'h-10 w-10'
      }
    },
    defaultVariants: {
      variant: 'default',
      size: 'default'
    }
  }
);

interface ButtonProps
  extends React.ComponentProps<typeof Pressable>,
    VariantProps<typeof buttonVariants> {
  className?: string;
}

const Button = React.forwardRef<
  React.ElementRef<typeof Pressable>,
  ButtonProps
>(({ className, variant, size, ...props }, ref) => {
  return (
    <Pressable
      className={cn(buttonVariants({ variant, size, className }))}
      ref={ref}
      {...props}
    />
  );
});

Button.displayName = 'Button';

export { Button, buttonVariants };
```

Responsive Design Patterns

```
const ResponsiveGrid: React.FC<{ children: React.ReactNode }> = ({ children }) => (
  <View className="
    grid
    grid-cols-1
    sm:grid-cols-2
    md:grid-cols-3
    lg:grid-cols-4
    xl:grid-cols-6
    gap-4
    sm:gap-6
    lg:gap-8
    p-4
    sm:p-6
    lg:p-8
  ">
    {children}
  </View>
);
```

```
const ResponsiveCard: React.FC<{ title: string; content: string }> = ({ title, content })
  <View className="
    bg-white dark:bg-gray-800
    rounded-lg shadow-sm
    hover:shadow-md
    transition-shadow
    p-4
    sm:p-6
    border border-gray-200
    dark:border-gray-700
  ">
    <Text className="
      text-lg
      sm:text-xl
      font-semibold
      text-gray-900
      dark:text-gray-100
      mb-2
      sm:mb-3
    ">
      {title}
    </Text>
    <Text className="
      text-sm
      sm:text-base
      text-gray-600
      dark:text-gray-400
      leading-relaxed
    ">
      {content}
    </Text>
  </View>
);
```

Testing and Quality Assurance

Testing Styled Components

```
// Button.test.tsx
import { render, fireEvent } from '@testing-library/react-native';
import { Button } from './Button';

describe('Button Component', () => {
  it('applies correct variant styles', () => {
    const { getByRole } = render(
      <Button variant="primary" testID="test-button">
        Test Button
      </Button>
    );

    const button = getByRole('button');
    expect(button).toHaveStyle({
      backgroundColor: '#3B82F6' // blue-500
```

```
    });
  });

  it('handles press interactions', () => {
    const mockPress = jest.fn();
    const { getByRole } = render(
      <Button onPress={mockPress} testID="test-button">
        Press Me
      </Button>
    );

    const button = getByRole('button');
    fireEvent.press(button);
    expect(mockPress).toHaveBeenCalledTimes(1);
  });

  it('applies custom className', () => {
    const { getByTestId } = render(
      <Button className="custom-class" testID="test-button">
        Custom Button
      </Button>
    );

    const button = getByTestId('test-button');
    expect(button.props.className).toContain('custom-class');
  });
});
```

Style Consistency Testing

```
// styles.test.ts
import { designTokens } from '../design-tokens';

describe('Design System Consistency', () => {
  it('maintains color contrast ratios', () => {
    // Test color combinations for accessibility
    const { colors } = designTokens;

    // Example: Test primary text on primary background
    expect(getContrastRatio(colors.brand.primary[^50], colors.brand.primary[^900]))
      .toBeGreaterThan(4.5); // WCAG AA standard
  });

  it('follows spacing scale', () => {
    const { spacing } = designTokens;
    const spacingValues = Object.values(spacing);

    // Ensure spacing follows consistent scale
    spacingValues.forEach((value, index) => {
      if (index > 0) {
        expect(parseFloat(value)).toBeGreaterThan(parseFloat(spacingValues[index - 1]));
      }
    });
```

```
  });
});
```

Migration and Best Practices

Migration from StyleSheet

Before (StyleSheet)

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#ffffff',
    padding: 16,
  },
  header: {
    fontSize: 24,
    fontWeight: 'bold',
    color: '#1f2937',
    marginBottom: 16,
  },
  card: {
    backgroundColor: '#f9fafb',
    borderRadius: 8,
    padding: 20,
    marginBottom: 12,
    shadowColor: '#000',
    shadowOffset: {
      width: 0,
      height: 2,
    },
    shadowOpacity: 0.1,
    shadowRadius: 3.84,
    elevation: 5,
  },
  button: {
    backgroundColor: '#3b82f6',
    paddingHorizontal: 24,
    paddingVertical: 12,
    borderRadius: 6,
    alignItems: 'center',
  },
  buttonText: {
    color: '#ffffff',
    fontSize: 16,
    fontWeight: '600',
  }
});

const Component = () => (
  <View style={styles.container}>
    <Text style={styles.header}>Welcome</Text>
    <View style={styles.card}>
```

```
      <Text>Card content</Text>
    </View>
    <Pressable style={styles.button}>
      <Text style={styles.buttonText}>Press me</Text>
    </Pressable>
  </View>
);
```

After (NativeWind)

```
const Component = () => (
  <View className="flex-1 bg-white p-4">
    <Text className="text-2xl font-bold text-gray-800 mb-4">
      Welcome
    </Text>
    <View className="bg-gray-50 rounded-lg p-5 mb-3 shadow-md">
      <Text>Card content</Text>
    </View>
    <Pressable className="bg-blue-500 px-6 py-3 rounded-md items-center">
      <Text className="text-white text-base font-semibold">
        Press me
      </Text>
    </Pressable>
  </View>
);
```

Performance Best Practices

1. Static Class Definition

```
// ✅ Good: Static classes
const BUTTON_CLASSES = {
  primary: 'bg-blue-500 text-white px-4 py-2 rounded-lg',
  secondary: 'bg-gray-200 text-gray-800 px-4 py-2 rounded-lg'
};

// ✅ Good: Conditional classes with static strings
<Button className={isLoading ? 'opacity-50 pointer-events-none' : ''} />

// ❌ Avoid: Dynamic class construction
<Button className={`bg-${color}-500`} />
```

2. Effective Content Configuration

```
// tailwind.config.js
module.exports = {
  content: [
    // ✅ Specific patterns
    './app/**/*.{js,ts,jsx,tsx}',
    './components/**/*.{js,ts,jsx,tsx}',
```

```
    // ✅ Include constants/design tokens
    './constants/styles.{js,ts}',
    './theme/colors.{js,ts}',

    // ❌ Avoid overly broad patterns
    // './**/*' - too broad

    // ✅ Exclude unnecessary files
    '!./node_modules',
    '!./**/*.test.{js,ts,jsx,tsx}',
    '!./**/*.stories.{js,ts,jsx,tsx}'
  ]
}
```

3. Build Optimization

```
// metro.config.js - Optimized configuration
const { getDefaultConfig } = require('expo/metro-config');
const { withNativeWind } = require('nativewind/metro');

const config = getDefaultConfig(__dirname);

// Optimize for production
if (process.env.NODE_ENV === 'production') {
  config.transformer.minifierConfig = {
    mangle: {
      keep_fnames: true,
    },
    output: {
      ascii_only: true,
      quote_keys: true,
      wrap_iife: true,
    },
    sourceMap: false,
    toplevel: false,
    warnings: false,
  };
}

module.exports = withNativeWind(config, {
  input: './global.css',
  inlineRem: 14, // Optimize for native rem size
});
```

Common Pitfalls and Solutions

1. Platform Differences

```
// ✖ Problematic: Assuming web behavior
<View className="overflow-scroll max-h-64">
  <Text>Long content...</Text>
</View>

// ✅ Solution: Use appropriate components
<ScrollView className="max-h-64">
  <View className="p-4">
    <Text>Long content...</Text>
  </View>
</ScrollView>
```

2. Flex Direction Confusion

```
// ✖ Problematic: Assuming web defaults
<View className="flex justify-between">
  <Text>Left</Text>
  <Text>Right</Text>
</View>

// ✅ Solution: Explicit flex direction
<View className="flex flex-row justify-between">
  <Text>Left</Text>
  <Text>Right</Text>
</View>
```

3. Safe Area Handling

```
// ✖ Problematic: Ignoring safe areas
<View className="flex-1 pt-12">
  <Text>Header content</Text>
</View>

// ✅ Solution: Use safe area components
import { SafeAreaView } from 'react-native-safe-area-context';

<SafeAreaView className="flex-1">
  <View className="pt-4">
    <Text>Header content</Text>
  </View>
</SafeAreaView>
```

Conclusion

NativeWind v4 represents the most comprehensive styling solution for React Native applications, providing the full power of Tailwind CSS while maintaining native performance characteristics. Its advanced features including CSS variables, platform variants, animations,

and responsive design capabilities make it suitable for professional-grade applications across all React Native platforms.

The combination of build-time optimization, runtime flexibility, and developer experience improvements positions NativeWind as the definitive choice for modern React Native styling. By following the patterns and best practices outlined in this documentation, developers can create maintainable, performant, and visually consistent applications that work seamlessly across iOS, Android, and web platforms.[3] [4] [1] [5]

⁂

1. https://www.nativewind.dev/blog/announcement-nativewind-v4
2. https://www.nativewind.dev/docs/tailwind/transitions-animation/animation
3. https://www.npmjs.com/package/nativewind/v/3.0.0-f09b31b.0
4. https://www.nativewind.dev/docs/getting-started/installation
5. https://www.nativewind.dev/docs