next¥app¥page.tsx

```tsx
import "./globals_variables.scss";
import Reference from './components/Reference/Reference';
import AllCodeView from './components/AllCodeView/AllCodeView'

export default function Home() {
  return (
    <>
      <Reference
        fontSize={28}
        text="テキスト"
        href="https://test.com"
        target="_blank"
        // onClickHandler={() => {}}
        isDisabled={true}
      />
      <AllCodeView
        fontSize={16}
      />
    </>
  );
}
```

next¥app¥globals_variables.scss

```scss
:root {
  --color-text-primary: #000000;
  --color-text-secondary: #fff000;

  --color-background-primary: #ffffff;
  --color-background-secondary: #0004ff;
}

// Next.js + CSS Modules では、.module.scss ファイル内では ローカルスコープのみ使用可です。
// body のようなグローバルセレクタは、通常の .scss ファイルか globals.scss に記述すべきです。
```

next¥app¥styles¥_mixins.scss

```scss
// @include mixins.font-size(16);
@mixin font-size($size, $base: 16) {
  $rem: calc($size / $base);
  font-size: #{$size}px;
  font-size: #{$rem}rem;
}

// @include mixins.font-weight(bold);
@mixin font-weight($weight) {
  $weight: (
    normal: 400,
    bold: 700,
    bolder: 900
  );
}

// title
@mixin title-normal {
  @include font-size(16);
  @include font-weight(bold);
  line-height: 1.5;
}
```

next¥app¥styles¥index.scss

```scss
@use "./_mixins";
@use "./_variables";

// このファイルの使用方法がわからない
```

next¥app¥components¥Reference¥Reference.tsx

```tsx
import ClassNames from 'classnames';
import Link from 'next/link'; // Link=<a>タグになる / Next.jsのLinkコンポーネント
import styles from './Reference.module.scss';

interface ReferenceProps {
  fontSize?: 12 | 14 | 28;
  text: string;
  /* リンク先のURL */
  href?: string;
  /* リンク先のURLが外部かどうか */
  target?: '_blank' | '_self';
  /* クリック時の処理 */
  onClickHandler?: () => void; // このように書くと、onClickHandlerがundefinedの時にエラーになるので、
  /* disabledかどうか */                                                    ?をつけてオプショナルにする
  isDisabled?: boolean;
}

const Reference: React.FC<ReferenceProps> = ({
  /* fontSizeは指定しないと14pxになる */
  fontSize = 14,
  text,
  href,
  target = '_blank',
  onClickHandler,
  isDisabled = false,
  }) => {
  const classNames = ClassNames({
    [styles.Reference]: true,
    [styles['Reference--disabled']]: isDisabled,
  });

  const classNamesFontSize = ClassNames({
    [styles[`Reference--fontSize${fontSize}`]]: fontSize,
  });

  const WrapperTag = href ? Link : 'div';

  return (
    <>
      <WrapperTag
        className={classNames}
        href={href as string}
        target={href ? (target === '_blank' ? '_blank' : '_self') : undefined}
        rel={href && target === '_blank' ? 'noopener noreferrer' : undefined}
        onClick={onClickHandler}
      >
        <h1 className={styles.Reference__title}>@mixinで設定：コンポネントに直で記述</h1>
        <div className={styles.Reference__textPrimary}>mixinsとvariables.scssで設定 | primary：{text}</div>
        <div className={styles.Reference__textSecondary}>mixinsとvariables.scssで設定 | secondary：{text}</div>
        <p className={classNamesFontSize}>module.scssで設定：{text}</p>
      </WrapperTag>
    </>
  );
};

export default Reference;
```

next¥app¥components¥Reference¥Reference.module.scss

```scss
@use "../../styles/mixins";

.Reference {
  display: flex;
  flex-direction: column;

  &__title {
    @include mixins.title-normal;

    @media (hover: hover) {
      &:hover {
        color: #ff0000;
        text-decoration: underline;
      }
    }
  }

  &__textPrimary {
    @include mixins.font-size(18);
    @include mixins.font-weight(bold);

    color: var(--color-text-primary);
    background-color: var(--color-background-primary);
  }

  &__textSecondary {
    @include mixins.font-size(23);

    color: var(--color-text-secondary);
    background-color: var(--color-background-secondary);
  }

  &--disabled {
    opacity: 0.5;
    // pointer-events: none;
  }

  &--fontSize12 {
    @include mixins.font-size(12);
  }

  &--fontSize14 {
    @include mixins.font-size(14);
  }

  &--fontSize28 {
    @include mixins.font-size(28);
  }

  // ▼ 仮で設定 ▼
  h1, div, p {
    margin: 10px 0;
  }
}
```