Design
Implementação
Criando um componente com styled components
Styled Components
Melhorando nossa estrutura
Utilizando o ThemeProvider
colors, fonts, units -> props
Como foi nosso desenvolvimento

Olá, eu me chamo hroad (apenas brincando e sem pesquisar no google, por favor) e eu vim contar um pouco sobre a utilização de styled-components com typescript e react native.

E vou contar um pouco sobre estes itens acima. Algumas coisas eu irei resumir o processo que nós passamos e até baseados em outros exemplos a fim de não ultrapassar o tempo que temos neste meetup e partimos para o código. E também pq nosso contexto, alguns detalhes acabam sendo bem amplos.

Mas vamos lá, acredito que umas das coisas que facilitam bastante a vida do desenvolvedor é ter softwares escaláveis, de manutenibilidade objetiva e que você precise ficar mais tempo procurando por algo do que realmente modificando e/ou consertando. E acredito que isto melhore com o tempo, estudo e prática. Mas é algo bem particular aqui.

Pensando nisso, não que eu vá defender o styled-components aqui ou qualquer outro item apresentado, mas vou trazer um pouco da minha visão nesse contexto.

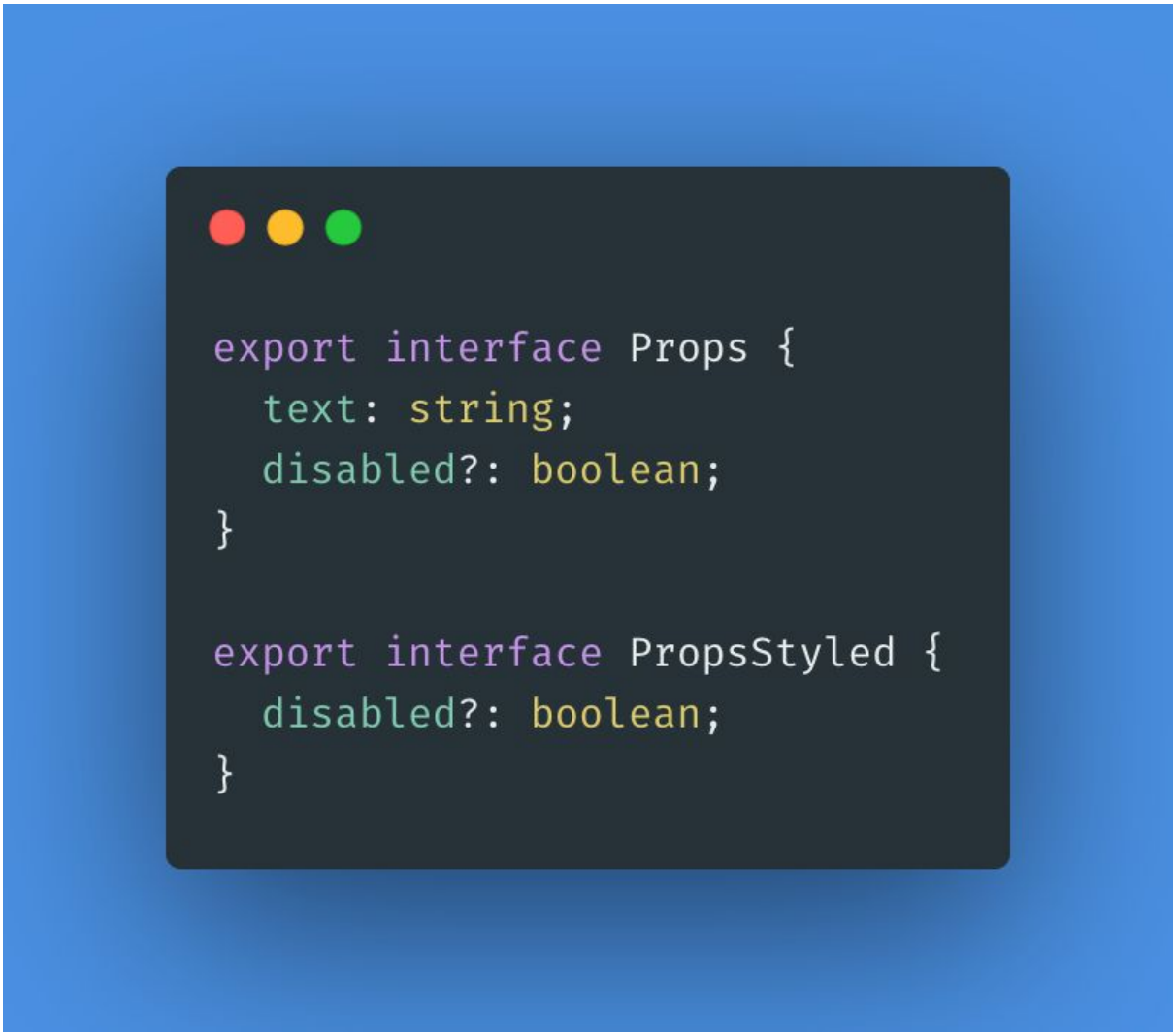
Olhando todos esses itens que eu citei acima e considerando aplicações web e mobile, um dos itens bens comentados é como componentizar a sua aplicação.

Pensando no nosso contexto e do nosso app, uma das primeiras coisas que desenvolvemos fora os componentes. Por quê? Bom, vamos imaginar o layout fictício abaixo. Imagina que eu começasse a dar ctrl+c, ctrl+v de uma tela para outra e depois alguém lá de design alterasse o laranja para verde. Quão custoso seria isto tanto para nós devs, quanto para a entrega deste item?

Agora imagina eu tenho um componente centralizado e reutilizável em qualquer parte do meu app. Eu tenho um único ponto de manutenção do meu código. Considerando a quantidade de vezes que eu irei ter que alterar essa informação já me parece bem mais razoável correto? Até o final, iremos melhorar ainda mais isto.
Ah, detalhe, não iremos entrar aqui no mérito de design system, tokens, etc, etc, pq seria um outro assunto ainda mais amplo.

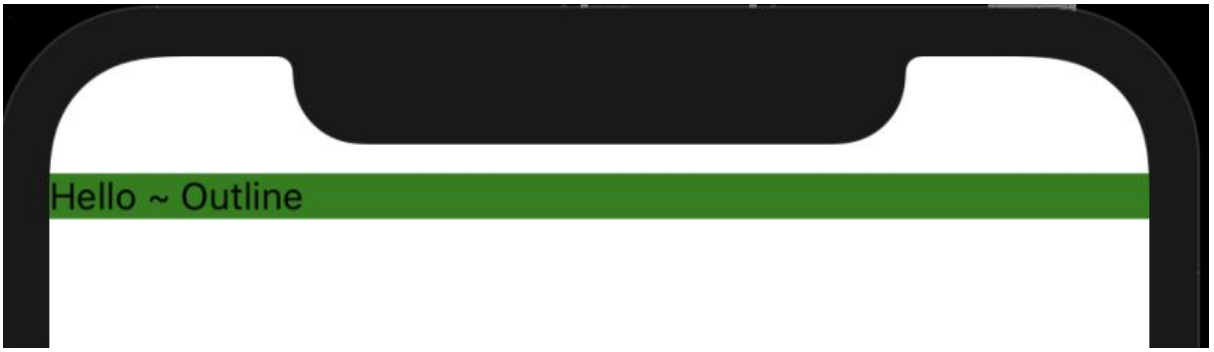
Bom, nós podemos criar componentes diretamente, sem styled components, typescript e etc, mas é justamente sua objetividade, seus benefícios que eu quero mostrar pra vocês.
Para criarmos um componente com styled, vamos desenvolver o seguinte:

```tsx
import React from 'react';

import styled from 'styled-components/native';

import { Props, PropsStyled } from './interface';

const Container = styled.TouchableOpacity<PropsStyled>`
  background-color: ${(props) => (props.disabled ? 'green' : 'blue')};
`;

const Label = styled.Text``;

const OutlineButton = ({ text, disabled }: Props) => {
  return (
    <Container disabled={disabled}>
      <Label>{text}</Label>
    </Container>
  );
};

export default OutlineButton;
```


export interface Props {
text: string;
disabled?: boolean;
}
export interface PropsStyled {
disabled?: boolean;
}


const App = () => {
return (
<SafeAreaView>
<OutlineButton text="Hello ~ Flat" disabled={true} />
</SafeAreaView>
)
}

```
const App = () => {
  return (
    <SafeAreaView>
      <OutlineButton text="Hello ~ Flat" disabled={false} />
    </SafeAreaView>
  )
}
```



Se vermos o resultado, nós extraímos do layout esse e podemos reutilizar ele em todo o cenário. E um item que eu quero trazer, é justamente aquele condicional dentro do nosso 'css'. Se fossemos desenvolver sem sc, faríamos algo como:

[disabled ? styles.x : styles.y]

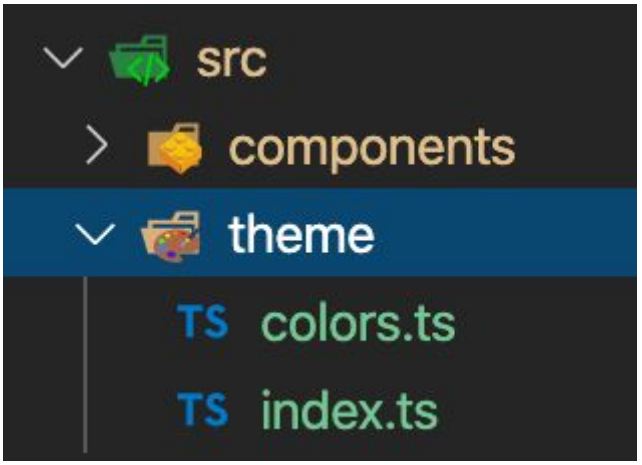
Imagina que eu tivesse loading, disabled, quantas coisas mais eu precisaria incluir aqui. Quantos imports eu precisaria fazer?
Na web, para quem tem mais este contexto, você não precisa mais de className. Ficar brigando com a nomenclatura. E quando fazemos alguma alteração nesse componente, ele acaba sendo refletido somente nele. Não teríamos o 'side-effect' ou efeitos colaterais, de alterar algo aqui e bugar todo nosso layout.

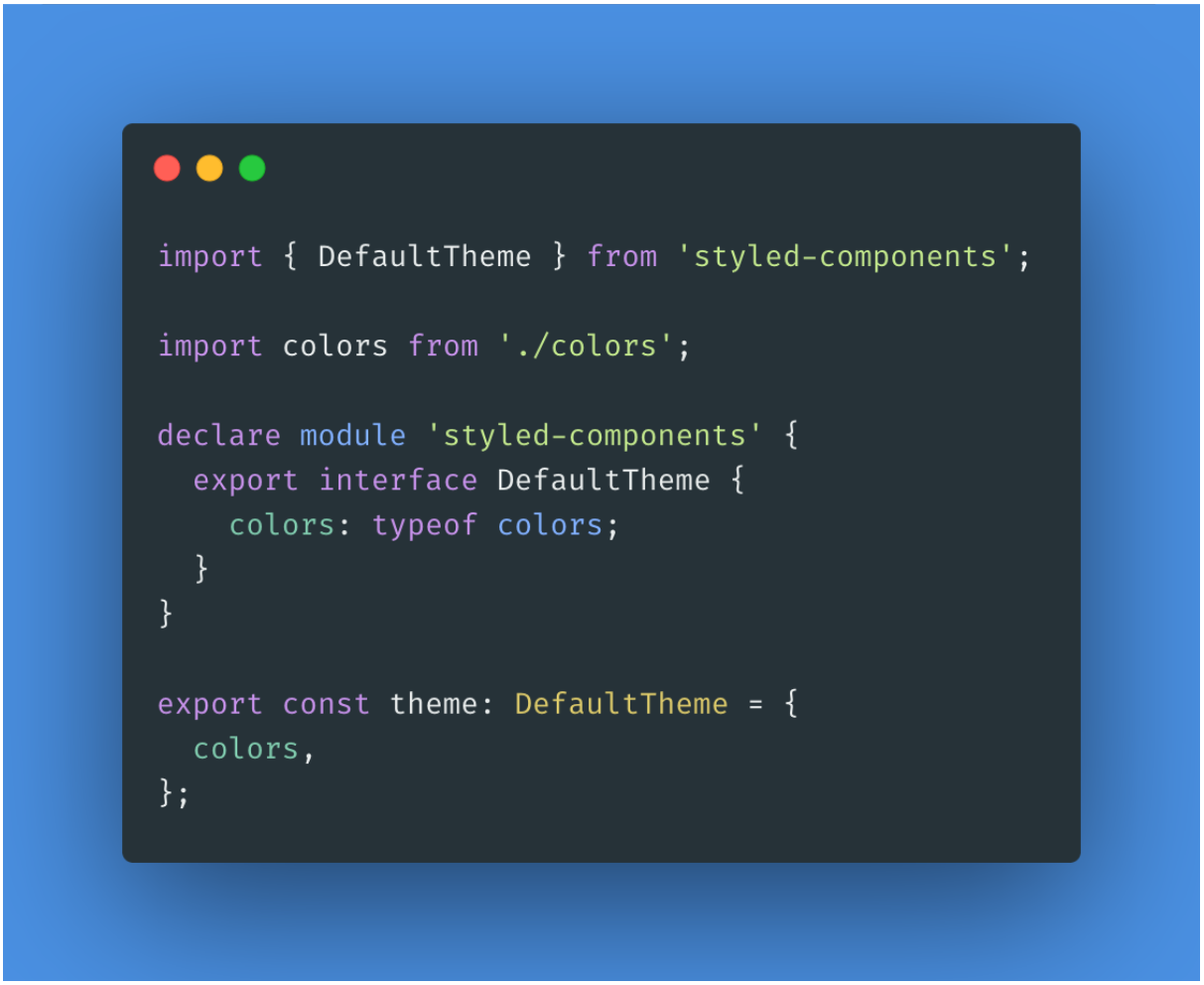
Quando eu olho para este ponto, não sei, me parece mais clara essa informação. O que este componente faz, quais ações ele pode vir a ter.

Então vamos fazer um recap, eu adiciono como dependência o styled components, crio qualquer componente utilizado styled.[nome-react-native] (se alguém quiser saber mais, pode pesquisar sobre template string) e o utilizo de maneira centralizada. Além disso, ele me permite realizar o que chamamos de css-in-js. Código css em arquivos javascript. Posso receber props para diferenciar por exemplo, um estado de disabled.

Mas helena, se considerarmos que você está passando a cor diretamente dentro do componente, ainda assim, considerando que eu teria que alterar a cor diretamente em cada componente, a manutenção não pode ser mais objetiva? Sim, e é exatamente isto que eu quero deixar como plus plus aqui.

Nós podemos utilizar o ThemeProvider que irá prover um tema para todos os seus componentes. Como fazemos isto?



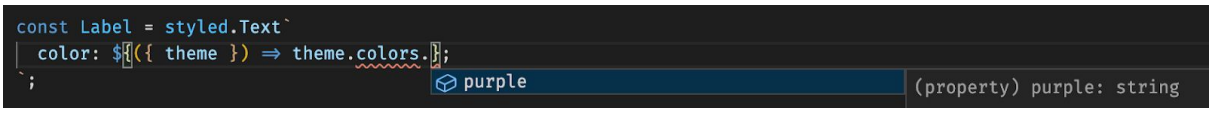


```ts
import { DefaultTheme } from 'styled-components';

import colors from './colors';

declare module 'styled-components' {
  export interface DefaultTheme {
    colors: typeof colors;
  }
}

export const theme: DefaultTheme = {
  colors,
};
```

src/theme/index.ts

src/theme/colors.ts

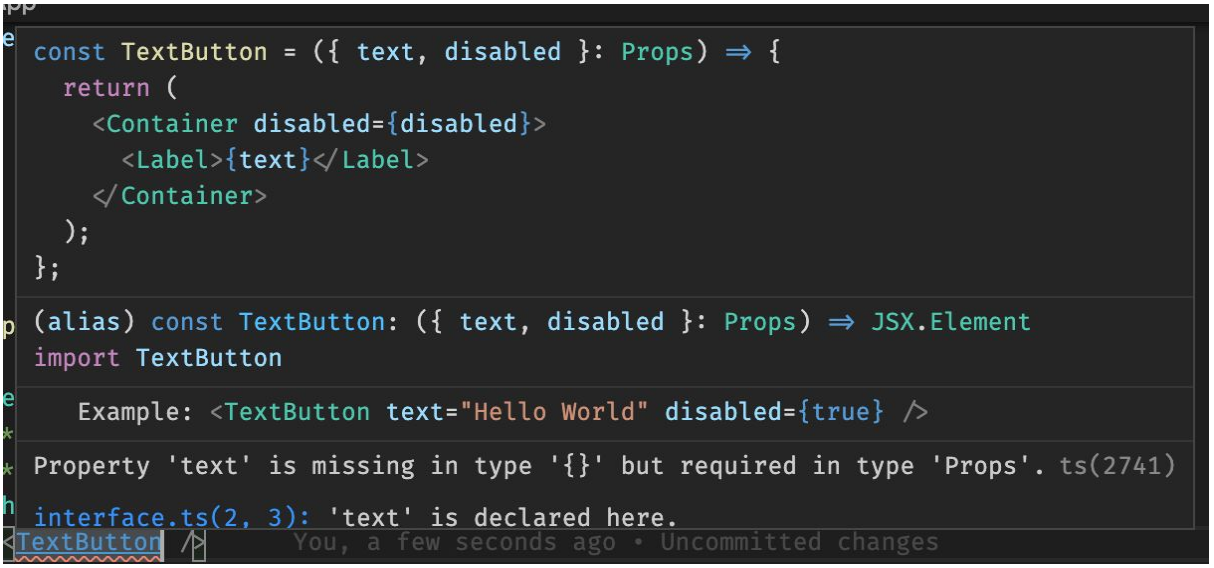
E como utilizo nos componentes?



Uau, está ficando um pouco maior meus componentes, nossa estrutura. Como eu sei o que cada componente irá receber? Quais props ele precisa receber? Você pode criar uma anotação em cima do componente a fim de facilitar a visualização e verificar tudo o que ele precisa para ser utilizado.

Incrível, não?

```tsx
/**
    ```
      Example: <TextButton text="Hello World" disabled={true} />
    ```
*/
const TextButton = ({ text, disabled }: Props) => {
  return (
    <Container disabled={disabled}>
      <Label>{text}</Label>
    </Container>
  );
};
```



```tsx
const TextButton = ({ text, disabled }: Props) => {
  return (
    <Container disabled={disabled}>
      <Label>{text}</Label>
    </Container>
  );
};

(alias) const TextButton: ({ text, disabled }: Props) => JSX.Element
import TextButton

  Example: <TextButton text="Hello World" disabled={true} />

Property 'text' is missing in type '{}' but required in type 'Props'. ts(2741)

interface.ts(2, 3): 'text' is declared here.
<TextButton />      You, a few seconds ago • Uncommitted changes
```

Maaaas, nem tudo são flores. O que eu senti desenvolvendo com styled components, typescript que ajuda e muito nossas tipagens e afins, foi um tempo bem maior no início da criação dos componentes. Por dois motivos: o primeiro fora o desafio de aprender mais a fundo sobre sc e o segundo fora que para a criação dos componentes, o tempo é quase uma montanha russa. No início ele sobe, sobe, sobe, mas depois para a criação das telas, esse tempo acaba diminuindo drasticamente.

No começo também, nós estruturamos os componentes e usualmente voltávamos para fazer alguma manutenção nele. E depois isso também diminuiu bastante.
Eles iam ficando mais robustos. Mas preparados para o nosso desenvolvimento.