これまでのところ、多くのことを行うことでウィジェットをカスタマイズできることがわかりました。各ウィジェットのプロパティを設定することもできます。そのため、車のウィジェットの色が黒の場合、単にその色プロパティを青に設定できます。

また、ウィジェットは不変であるため、古いウィジェットは破棄され、更新されたプロパティで新しいウィジェットが作成されます。

また、スタイルや、Flutterによって事前定義されている他のプロパティを変更することもできます。

スライダーウィジェットの場合、activeColor、分割などを変更できます。ただし、変更できるプロパティはFlutterチームの事前計画に依存しているため、変更できるプロパティの数は限られています。一般的に必要。

したがって、さらに制御したいときにさらに掘り下げたい場合は、SliderThemeDataに進み、disabledActiveTickMarkColorなど、より具体的なテーマを設定することができました。

しかし、その後、私たちも行き止まりになりますか？

より多くのカスタマイズが必要で、カスタムウィジェットをゼロから作成したい場合はどうでしょうか？多くのプログラミング言語では、SwiftまたはJavaから来た場合、またはiOSまたはAndroidの開発を行ったことがある場合、一般的にこれを行う方法は継承です。

そのため、スライダーなどのコンポーネントを継承し、それらのコンポーネントのプロパティまたはメソッドの一部をオーバーライドします。

そして、これは一般的にAndroidとiOSの両方で私たちが行う方法です。しかし、Flutterでは、その構築方法は継承よりも合成を優先します。

それはどういう意味ですか？

これは、可能な限り単純なウィジェットからゼロから物事を構築しようとすることを意味し、そのようにしてFlutterはコンポーネントのパフォーマンスを維持し、アプリを高速に保ちます。では、実際にウィジェットを作成するにはどうすればよいのでしょうか？

コンテナなどの基本的なウィジェットについて考えても、実際には、ConstrainedBoxを備えたLimitedBoxや、DecoratedBoxを備えたPaddingを備えたLimitedBoxなどの小さなウィジェットで構成されています。そのため、非常に基本的で非常に単純なことを行うウィジェットの集まりが集まって、コンテナを構築しました。

そしてこれは、レゴの小さな破片のような非常に単純な不変の構成要素を持つという考えに戻ります。

そして、これらのシンプルなウィジェットを組み合わせることで、より複雑でより興味深いウィジェットを構築できます。

したがって、FloatingActionButtonなどの事前に構築された大きなウィジェットを使用して、何らかの方法でそれを継承してから特定のことをオーバーライドしようとするのではなく、Flutterでは、それを小さな断片に分割して、オープンソースコードを作成し、Flutterからアクセスできる最も単純で最も基本的なビルディングブロックを使用して、自分でコードをビルドします。そして、それがまさにこのレッスンでやろうとしていることです。大丈夫。

そのため、アイコンカード、​​スライドカードは既に作成済みです。

これで、作成する最後の行だけができました。

それでは、ここでそれを行ってみましょう。

そのため、現在2枚の再利用可能なカードがあり、それらにいくつかの子供を与える必要があります。

最初のcarChildにcarChildを追加しましょう。

そしてこの場合、上から下に向かって多くのものを積み重ねなければならないので、列を追加するのも理にかなっています。そして、このコラムでは、メインのAxisAlignmentをすでに中央に設定します。

このように、すべてが垂直である主軸に沿って列の中心にまとめられます。

そして、私たちはコラムに子供たちを与えます。

そのため、最初のものは単に「WEIGHT」と言うテキストウィジェットになります。

この場合、これまで使用してきたスタイルと同じスタイル、つまりklabelTextStyle定数を使用します。

これで、列の中央にテキストが画面に表示されるようになります。

次は、ユーザーが入力しようとしている体重の実際の数値を追加します。

そのため、これを追跡する変数が必要になります。

したがって、高さのすぐ下に追加し、整数としても保持します。

そして、私たちの体重は、おそらく60キロと言って始めようとしている、それは大丈夫です。

それで、変数としての重みの数値を取得しました。

したがって、高さの調整と同じように、別のテキストウィジェットを追加できます。

ここにデータとして追加します。

そのため、重みを設定し、それを文字列に変更します。それがテキストウィジェットで必要なものだからです。

それが得られるものです。

今、私たちの体重もスタイリングされます。

しかし、これはnumberTextStyleを使用してスタイル設定されます。これは、これに非常に似ている、非常に大きな太字の外観です。

これで、体重と開始体重が決まりました。

したがって、ユーザーが実際に体重と年齢を切り替えることができるように、ボタンを追加するだけです。

これらのボタンは、FloatingActionButtonsを使用したときに以前から見てきたように、これと非常に似ていますか？中央にアイコンの子がある丸いボタン。

そこで、ここにFloatingActionButtonを追加すると、FABを短く使用して、信頼できるAndroid Studioが必要なものを取得できるようにします。

そして今、先に進んでEnterキーを押して保存すると、FloatingActionButtonがポップアップ表示され、実際に似ていることがわかります。

私がする必要があるのは、おそらく子供にそれを与えることです。子はおそらくアイコンになり、アイコンのデータはアイコンの1つを使用するだけで、addを使用することができます。

真ん中にアイコンが付いた丸いボタンがあり、暗いテーマを使用しているためこの色でスタイル設定されていますが、backgroundColorを変更することで色を変更するのも簡単です。そのため、色を明るいグレーに変更したり、0xFFのように完全に不透明な4C4F5Eにすることができます。

これが、これまでの成果です。

そのアイコンを白い色に変更しましょう。

そこで、アイコンウィジェットに移動して、便宜上、色をcolors.whiteに変更します。ここで2つが必要なので、行が必要です。

そこで、FloatingActionButtonを行ウィジェット内にラップし、もう1つ追加します。

そして、ここのすぐ下に貼り付けます。たぶん、sizeBoxを追加して、間に少しスペースを空けます。多分10の幅を与えます。

それを試して、それがどのように見えるか見てみましょう。

大丈夫。

それはかなりきれいに見えます。

私たちがする必要があるのは、メイン軸に沿って中央に行を変更するだけで、今は真ん中に配置する必要があります。

大丈夫。

よさそうだ。

しかし問題は、FloatingActionButtonのドキュメントを実際に読んだ場合、画面ごとにせいぜい1つのフローティングアクションボタンを使用することを明確に示していることがわかります。

そして、彼らには特定のユースケースがあります。

今ではドキュメントがそれに対してそれを推奨しているだけでなく、違う見た目が欲しい場合はどうでしょうか？

フローティングアクションボタンのカスタムシェイプが必要な場合、または希望どおりに表示する場合はどうなりますか？

これまでのところ、ほとんどのFlutterコンポーネントをそのまま使用しており、Flutterウィジェットの組み合わせから独自のウィジェットを作成するために、さまざまな方法でコンポーネントを組み合わせようとしています。マテリアルから使用してきたウィジェットのテーマを深く掘り下げ、その特定の部分を更新して、念頭に置いたデザインのようにします。

しかし、このレッスンでは、Flutterは完全にオープンソースであるため、これらの各ウィジェットのコードを実際に掘り下げる方法を示したいと思います。そのため、フローティングアクションボタンやスライダーを見ているコンポーネントはすべて、すべてのソースコードを参照してください。iOSなどと言うことはできません。 iOSボタンがどのように作成されるかは実際にはわかりません。

ソースコードを確認できることの利点は、CONTROLまたはCOMMANDを押しながらフローティングアクションボタンをクリックすると、floating\_action\_button.dartファイルが表示されることです。

このファイルには、フローティングアクションボタンを作成するために必要なすべてのコードが含まれています。

そのため、メインのビルドメソッドを検索すると、このビルド方法がわかります。

さあ、行きましょう。フローティングアクションボタンのビルドメソッドは次のとおりです。メインアプリテーマからテーマデータをコピーし、前景色またはアクセントテーマを設定してから、実際のボタンを作成します。フローティングアクションボタンは、RawMaterialButtonと呼ばれるFlutterの基本コンポーネントから作成されていることがわかります。そしてこれには、押されたときに何をすべきか、その高さなど、多くのプロパティがあり、さらに深く掘り下げることができます。

したがって、CONTROLまたはCOMMANDを押したままにすると、原料ボタンに移動し、独自のビルドメソッドをチェックアウトして、このビルド方法を確認します。

次に、タップを検出できるInkWellを持つconstrainEDboxを使用して構築されていることがわかります。

これは、原料ボタンがボタンであるためです。そのため、タッチを検出する必要があります。

しかし、それがまったく必要ない場合は、単にマテリアルを作成できます。そして、controlキーを押しながらマテリアルをさらにクリックすると、メインのmaterial.dartページに移動し、そのビルド方法を見ると、これもいくつかの小さなコンポーネントからビルドされていることがわかります。

ウサギの穴をどれだけ深く掘り下げたいかに応じて、必要に応じてカスタムのデザインまたはコンポーネントを実際に作成できます。デフォルトのFlutterコンポーネントにまったく依存する必要はありません。

また、Flutterのパッケージを見ると、カスタムスライダーなどの独自のカスタムアイテムを作成したときに作成したパッケージがたくさんあることがわかります。

たとえば、このスライダーを見ると、この人が2つの親指の範囲のスライダーを構築していることがわかります。そして、あなたは彼らがこのウィジェットをどのように構築したかを教えてくれることを見ることができます。

彼らは基本的に、このカスタムウィジェットを表示するために、ゼロから純粋な図面として作成しました。

ですから、私が言いたいのは、Flutterが提供するウィジェットは非常に便利であり、ほとんどの場合、必要なものはすべて揃っているということです。

ただし、特に実装したいデザインがある場合は、デフォルトのウィジェットの形状や色、スタイルに制限する必要はなく、単に独自のウィジェットを作成するだけで済みます。

それでは、独自の原料ボタンを作成してみましょう。また、フローティングアクションボタンに依存する代わりに、行内に作成してボタンとして機能させることができます。そのため、一番下のショートカットを使用して、新しいステートレスウィジェットを作成します。そして、ウィジェットをRoundIconButtonと呼びます。

だから、できる限り説明的にしようとするつもりです。

これで、私の丸いアイコンボタンは、フローティングアクションボタンの作成方法と非常によく似た方法で作成されます。

また、原材料ボタンにも依存します。

それでは、RawMaterialButtonを返しましょう。

そして、この場合、塗りつぶしの色、強調表示の色、スプラッシュカラー、または標高などを含む多くのプロパティがありますが、それらはすべてデフォルトで提供されているので、設定するだけです。実際に変更したいそれでは、原料ボタンのプロパティを設定してみましょう。

最初に、おそらく形状が必要になります。そして、shapeプロパティがShapeBorderであるものを期待し、デフォルトでRoundedRectangleBorderが与えられていることがわかります。

これは私たちが望んでいるものではないかもしれないので、他にどのような形状があるのか​​見てみましょう。

したがって、ドキュメントを見ると、これがシェイプアウトラインの基本クラスであり、複数の境界線を一緒に追加する方法を処理し、さまざまなシェイプを定義できることがわかります。

たとえば、丸い長方形に円またはRoundRectangleBorderが必要な場合は、circleCorderを使用できます。これらのさまざまなスタイルの境界線をすべて選択できます。

したがって、この場合、単に円の境界線を選択して円として維持し、次に原材料ボタンに塗りつぶしの色を追加し、単にその色の上にコピーします前から持っています。

これで、フローティングアクションボタンを使用していた場所に簡単に移動でき、代わりにカスタムの丸いアイコンボタンを使用できます。

それでは、どのように見えるかをチェックしましょう。

そのため、フローティングアクションボタンよりも少し小さく、高さはありません。

周囲に影はありません。

現時点では完全にフラットです。

それを変更して、見た目通りに再描画しましょう。

サイズを更新してみましょう。

さて、どうやってそれを行うのか知っていますか？

すべてのドキュメントを読んで、変更する必要があるプロパティを確認できます。またはさらに簡単に、フローティングアクションボタンを見て、Flutterチームが実際に原材料を使用してこのボタンを作成した方法を確認できます

ボタン。これらのサイズの制約があることがわかります。

そして、このファイルでこれを見つけようとすると、それがここにあり、いくつかの定数、最小サイズの制約、通常のサイズの制約があることがわかります。

したがって、これらの定数が何であるかを見ると、実際には、56の幅と56の高さに対して厳しいボックス制約を持つように設定されていることがわかります。

では、なぜマテリアルボタンの制約としてこれを実装するのでしょうか？

そのため、制約を追加し、Flutterフローティングアクションボタンから非常に便利にコピーしたボックス制約を設定できます。

保存をクリックして、形状がFAB形状とまったく同じであることを確認しましょう。それでは、素材ボタンに標高を追加し、約6ピクセルの標高を追加します。

しかし今、ボタンを見ると、まだ完全に平らであることがわかります。

実際には影はなく、ズームインして確認することもできますが、実際にここにあるこのフローティングアクションボタンとは異なり、追加される標高はありません。

それはなぜですか？

それは、すべてのマテリアルボタンにonPressedパラメーターがないため、タッチに応答できない場合、実際には無効状態にあるためです。

したがって、無効化された標高の設定もあります。

それを6として追加すると、標高を取得できることがわかります。しかし、それよりも優れているのは、実際に単にonPressedを追加し、誰かが丸いアイコンボタンを押したときに何が起こるかを定義することです。

ここで保存をクリックすると、私の影がフローティングアクションボタンとまったく同じであることがわかります。

制約を変更することと、マテリアルボタンが無効か有効かによってonPressedが変更されるという事実をどのようにして知りましたか？

もちろん、それはドキュメントです。

したがって、ボタンサイズを定義するには、制約を変更することをここで述べました。また、高さは、ボタンが有効になっているが押されていない場合に使用されます。したがって、ボタンはデフォルトで無効になっています。ボタンを有効にするには、onPressedプロパティをnull以外の値に設定します。

したがって、ここには本当に魔法はありません。

すべては、ドキュメントを本当に注意深く読むことに由来しています。さて、マテリアルボタンに必要な他の唯一のものは子であるため、そこにアイコンを入れることができます。

そして、アイコンは左右で異なります。

左側がマイナス、右側がプラスになります。

したがって、幸運にも、素材ボタンには子プロパティがあり、これを設定できます。これを丸いアイコンボタンに渡します。

したがって、最終プロパティとして設定し、ウィジェットとして追加するだけで、子と呼ばれます。

そして、丸いアイコンボタンを初期化するとき、this.childを子プロパティに渡す必要があります。そして、ここに移動して、原料ボタンに割り当てます。

これをそのままにして、いくつかのアイコンデータを含むアイコンを渡すことができますが、これを少し簡単にすることもできます。これを丸いアイコンボタンと呼ぶことを考えると、ほとんどの場合、おそらくそれは周りにあり、おそらくアイコンを含むでしょう。

では、なぜ子供をアイコンウィジェットに変更し、アイコンデータのみを渡す必要があるのでしょうか。

プロパティをウィジェットからIconDataに変更し、ここでそれを渡すことができるようにアイコンを呼び出します。アイコンウィジェットとそのすべてのファミリーを持つ子全体ではなくアイコンを初期化します。

丸いアイコンボタンに移動して、FontAwesomeアイコンに変更するアイコンを指定し、プラス記号を使用します。これは、少し太くて見やすいためです。

これがここにあるものです。これをコピーして、フローティングアクションボタンを削除することもできます。

そして、これをマイナスに変更して、左側にマイナス、右側にプラスを設定します。

だから、かなりきれいに見えます。

そのため、独自のバージョンのフローティングアクションボタンを作成しましたが、現在はフローティングアクションボタンとほとんど同じように見えます。しかし、もちろん、多くのコンポーネントを実際に変更できる基本的な基本的な構成要素を使用しているため、これを無限にカスタマイズできます。

たとえば、形状であるため、沼地の標準的な円だけが必要ない場合は、たとえば丸い長方形に変更できます。

したがって、角丸長方形の境界線を選択し、内側で境界線半径を変更して境界線半径の円形を使用できます。

そして、そこにおそらく半径を提供できます。

そして今保存ボタンを押すと、ボタンとして丸い長方形が表示されます。

または、楕円、楕円、またはカスタムシェイプに変更することもできます。また、標高を変更することもできます。

影のある高台のものの見方が気に入らなければ、完全に平らにすることができます。

したがって、これらのデフォルトのFlutterウィジェットに限定されていると感じることはありません。

Flutterの世界は本当にあなたのカキであり、画面に好きなものを作成できます。だから私はこれをサークルに戻すつもりです。サークルとしてとても気に入っているからです。また、私は実際に完全に平らにしたいので、実際に標高をゼロに変更します。

私の目にはずっと良く見えますが、それはあなた次第です。

任意の方法で変更できます。

しかし、実際には、丸いアイコンボタンに少し機能性を与える時間です。

重みを更新するアクションを実行するために、onPressedに何かを渡す必要があります。

これで高さを使用してこれを実行しました。

チャレンジとして、ここで体重カードを更新して、プラスボタンをクリックすると、その数が増えるようにします。

そして、マイナスボタンをクリックすると、数字が減ります。

セット状態はあなたの友達です。挑戦してみてください。

OK。

そのため、スライダーで以前に行ったように、ここでset numberを使用してその番号を更新する必要があります。

そのため、数値は現在weightと呼ばれる変数に格納されていることがわかります。

そして、ユーザーがこれらのボタンのいずれかをクリックするたびに更新する必要があります。

丸いアイコンボタンの内部で、onPressedのような別のプロパティを渡す必要がありますか？

しかし、現時点では明らかにそのプロパティはまだありません。

それでは、最終関数のデータ型をonPressedとして追加してみましょう。これもコンストラクタに追加します。

したがって、this.onPressedになります。

今、あなたがそれを何と名付けるかは本当に重要ではありません。現時点では、

onPressedの原料ボタンの名前と同じにしておき、単にここに渡すだけです。

しかし、命名はあなた次第です。

また、ウィジェットは丸いアイコンボタンと呼ばれるため、必要に応じてアイコンとonPressedの両方を必要に応じてマークします。

したがって、これら2つを必要に応じてマークするためのロジックです。

丸いアイコンボタンを作成するときに、これらのプロパティのいずれかが追加されていない場合は警告が表示されるという事実以外は何も変わりません。

そのため、行の最初の左側にあるonPressedの場合、これはマイナス記号であるため、これを減らすことで体重を更新したいです。

そのため、セット状態を呼び出し、セット状態の内側で体重プロパティを取得し、1を減算します。

そして、その略記は-です。

そして、プラス記号の付いた丸いアイコンボタンで、onPressedメソッドは体重を増やします。

そのため、その状態を呼び出し、++を追加して重みを増やします。

そのため、丸いアイコンボタンを押すたびに1つずつ追加されます。

だから今、私たちがしなければならないのは、保存してヒットすることです。

だから今マイナスを打った場合、ダウンしてプラスになります。

その完璧な。

あとは、最後のカードである年齢カードを実装するだけです。

だから、あなたが知っていることを知り、あなたが学んだことを使って、挑戦としてこれを試して完了してください。

そして、私はここで解決策をあなたを待っています。

大丈夫。

したがって、年齢カードは体重カードとほとんど同じに見えます。

したがって、重量カードの作成方法はすでにわかっています。

そして、その重量カード内のこの丸いアイコンボタンも再利用します。

ここに最後のカードがあります。これをカードの子にしましょう。

そして、これには列が含まれます。列の主軸は中央に配置され、「AGE」という単語が含まれるテキストウィジェットを含むいくつかの子が表示され、ラベルテキストスタイルの定数であるスタイルが設定されます。

そして、そのテキストウィジェットの後に、年齢の実際の数を含む別のテキストウィジェットを作成します。

そのため、変数を一番上に作成する必要があります。

また、整数にすることもできますが、おそらく20であっても、実際には問題ではありません。

ここで、その年齢をテキストとして含めますが、もちろん、整数から文字列に変換する必要があります。そして、年齢のスタイルは、その大きなフォントサイズであるKNumberTextStyleになります。

だから、私たちのデザインをチェックして、すべてが期待どおりに見えることを確認しましょう。

そして今必要なのは、2つのボタンを含む行です。

行を追加して、コンテンツをメイン軸に沿って中央揃えしましょう。

以前とまったく同じ方法で。そして、2つのボタンがあります。

したがって、2つの子を追加し、丸いアイコンボタンを使用します。丸いアイコンボタンには2つのプロパティが必要です。

したがって、Enterキーを押すとすぐに追加されます。

そして、左のアイコンはFontAwesomeIcons.minusになります。そして、私たちが何をしたいのかが押されたら、状態を設定し、年齢を1つ減らしたいと思っています。

次に、2つのボタンの間隔を空けることができるように、サイズのボックスを追加しましょう。

そして、幅を10に追加してから、別の丸いアイコンボタンを追加するだけです。そして、このアイコンはプラス記号になるので、.plusを押して、状態を設定し、年齢に追加します。

だからage ++。今やらなければならないことは、エラーがあるかどうかを確認することです。ここにエラーがあります。これは、コンマを忘れたためです。

今、保存をクリックすると、Dartはコードを再フォーマットし、アプリをチェックアウトできます。

だから、体重、年齢、男性と女性の身長があり、インターフェイス全体がほぼ完成しています。

ですから、カードをクリックして性別を選択できます。

スライダーを動かして身長を選択できます。

それを私の身長に乗せてみましょう、それから私の体重は、ええですが、63歳で、私の年齢は秘密です。

これで、このユーザーインターフェイスで受信したすべてのデータからBMIを計算する準備が整いました。

そしてそれを行うには、2番目の画面を作成してそこに移動できる必要があります。

マルチスクリーンアプリを作成するのは今回が初めてで、どのように学習するのか

次のレッスンでそれを正確に行うために。

そこで会いましょう。